Estudio de la endogamia de los circuitos booleanos

Research into the endogamy of boolean circuits

Enrique Román Calvo

DOBLE GRADO EN INGENIERÍA INFORMÁTICA - MATEMÁTICAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo de fin de grado - Curso 2019/2020

Fecha: 26 de junio de 2020

Directores:

Narciso Martí Oliet
Ismael Rodríguez Laguna

Índice general

Agradecimientos

Agradecer es sencillo, solo debes fingir ese sentimiento, más el sentimiento de agradecimiento no es fácil de fingir. Por ello, todo lo que bajo estas líneas se recoge es un reconocimiento explícito a aquellas personas sin las que este trabajo no hubiese salido adelante.

Así, en primer lugar debo agradecer la infinita paciencia y el meticuloso trabajo de mis dos directores, Narciso Martí Oliet e Ismael Rodríguez Laguna, que me han ayudado y guiado de una manera excelente en el trabajo y han sido unos defensores a ultranza de este proyecto con auténtico fervor por lo que repetiría esta misma experiencia junto a ellos sin dudarlo si pudiera volver atrás. Uno de sus mayores logros ha sido el saber delegar a tiempo el asesoramiento en terceros cuando ha procedido. En concreto, quiero agradecer especialmente a Rubén Rafael Rubio Cuéllar por su disposición a ayudarme con aspectos técnicos de C++, Automake y Linux y a enseñarme cómo hacerlo de manera rápida y elegante a partes iguales; así como a Javier Rodríguez Laguna por dejarme usar una máquina virtual del Instituto de Física Teórica de la UAM para poder ejecutar los experimentos realizados.

Además, quiero agradecer a mis padres el apoyo mostrado durante estos años de carrera, ya que de no ser por ellos, por su apoyo económico y emocional y los valores que me han transmitido no hubiera sido la persona que soy y no estaría hoy escribiendo estas líneas que estás leyendo.

Por una parte, sin el apoyo de mis amigos no habría podido entregar en tiempo y forma este trabajo, me habría desesperado en el proceso. Así, por todas las imágenes no deseadas de *manteles* y *espaguetis* que habéis visto estos meses: Lucía, Isabel, Iván, Irene... gracias. Gracias inmensas a mis compañeros de clase, que partieron siendo eso y acabaron siendo mi vida; unos buenos y necesarios amigos que os recomiendo a todos encontrar alguna vez en vuestra vida. A todos y cada uno de ellos, gracias. Pero, sin dudar a dudas, debo agradecerle este trabajo a Francisco Bellot Rosado, gran matemático y gran persona que me ha dado grandes valores y amigos que son valores. María, Antonio, Ruth, Victoria... sois vosotros.

Por otra parte, debo también agradecer a Carmen y María José que con su dedicación y su entusiasmo me demostraron que la docencia también puede ser vocacional y destacaron para mí de manera sobresaliente como profesoras en la etapa de educación obligatoria. Vocación que afortunadamente han demostrado la mayor parte de los profesores que en esta universidad me han impartido y eso verdaderamente se agradece. Desgraciadamente, también debo mencionar al XXI ENEM haberme consumido todo el tiempo libre que este trabajo me ha dejado, pero haberme brindado a su vez la oportunidad de conocer a gente maravillosa y muy trabajadora. Gracias chicos por este congreso que casi hicimos pero no pudo ser.

Y por último y más importante, quiero agradecerte a ti, lector, por no haber imprimido este documento y leerlo en su formato digital. Gracias por no malgastar papel de manera innecesaria. Y sobre todo y más importante, gracias por leerlo.

Resumen

El objetivo de este trabajo consistió en desarrollar un experimento que permita analizar múltiples tipos de circuitos booleanos y las funciones que estos computan, de manera que podamos analizar las relaciones entre ambos conceptos.

En el primer capítulo, se expone una definición formal del concepto archiconocido de circuito booleano, se analiza el crecimiento doblemente exponencial del conjunto de circuitos con m bits de entrada y se dota de un orden total a dicho conjunto. En el segundo capítulo, se demuestran una serie de resultados sobre vectores con determinadas propiedades que nos permiten recorrer de manera eficiente el conjunto de circuitos booleanos que tienen determinada profundidad y anchura. Este algoritmo se ha implementado en C++ utilizando técnicas de programación concurrente y estructuras de datos adecuadas para garantizar la eficiencia del mismo.

A lo largo del tercer capítulo se exponen distintas métricas para medir la endogamia de los circuitos booleanos, así como intuiciones que justifican estas definiciones. Además, se incorpora un breve cuarto capítulo que versa sobre un tipo de gramática libre de contexto particular que genera funciones booleanas y su correspondencia con los circuitos booleanos.

Finalmente, este trabajo concluye con un quinto y último capítulo en el que se analizan los resultados obtenidos en este experimento y se exponen las conclusiones consecuentes.

Palabras clave:

Complejidad de circuitos. Circuito booleano. Endogamia de un circuito. Función booleana. Índice de un circuito. P/poly. Propiedad del buen prefijo. Clase de complejidad. Algoritmo del índice. Correlación entre funciones y circuitos.

Abstract

The aim of this project was to develop an experiment that allow us to analyze different kinds of boolean circuits and the functions they compute to study the relationship between both concepts.

In the first chapter, a formal definition of boolean circuits is introduced, the double exponential growth of the set of circuits with m bits of input is analyzed and a total order in this set is defined. During the second chapter, some results about vectors of some kind are proved, which allow us to efficiently sweep the set of boolean circuits with fixed depth and width. This algorithm has been implemented in C++ using concurrent programming and adequate data structures to ensure its efficiency.

During the third chapter, some metrics to measure the endogamy of boolean circuits are defined and some ideas beyond these definitions are added. In addition, there is a fourth brief chapter where the relationship between a special type of context-free grammar and its equivalence with our boolean circuits are discussed.

To conclude, a final fifth chapter is added to analyse the results obtained in the experiment and the conclusions these results led us to.

Keywords:

Circuit complexity. Boolean circuit. Endogamy of a circuit. Boolean function. Index of a circuit. P/poly. Good prefix property. Complexity class. Index algorithm. Correlation between functions and circuits.

Introducción y motivación

Cuando uno se pregunta para qué sirve estudiar un circuito formado por puertas lógicas, lo primero que se le viene a la cabeza es toda la teoría de la arquitectura de computadores que nos dice que, *en esencia*, un computador es un conjunto de puertas lógicas conectadas entre sí. Así, el estudio de los mismos puede proporcionar importantes mejoras en el rendimiento de nuestros computadores. Por ejemplo, ese es el caso de los *grafos and-inversor*; una manera de modelizar los circuitos lógicos que es útil en distintos contextos relacionados con el hardware como en el expuesto en este artículo [1].

Sin embargo, no será ese el camino que tomemos en este documento. Imaginemos por un momento un circuito lógico formado por puertas and, or de dos entradas y not de una entrada en el que no hay retroalimentación, es decir, es un circuito puramente combinacional; el cual está conectado a n inputs. Dicho circuito, dependiendo de cuál sea el valor de su entrada, devuelve una salida binaria, 0 o 1. Además, es claro que hay 2^n combinaciones de valores distintos que pueden tomar los n inputs: cada uno de ellos solo puede tomar el valor $\top$ o el valor $\bot$. Así, podemos representar una función booleana con 2^n bits, cada uno de ellos asociado a la evaluación de la función en cada uno de las combinaciones de valores que los inputs pueden tomar. Por otra parte, es claro que toda función es computable con un circuito booleano: nos basta para cada conjunto de valores del input que den 1 realizar el and de los mismos y luego juntarlos todos con un or. Pero... ¿cuántas de estas funciones son computadas con un circuito que utilice un número de puertas polinómico en n?

Eso ya no es tan sencillo de averiguar. Tal y como se exhibe en el libro *Computational Complexity - A Modern Approach* [2], para cada valor de n existe alguna función booleana que no es computable por ningún circuito de tamaño $\frac{2^n}{10n}$. De hecho, la gran mayoría de dichas funciones no son computables por ningún circuito de tamaño polinómico, ya que, siendo p el polinomio que acota el número de puertas del circuito, el número de circuitos posibles está en $\mathcal{O}(p'(n)^{p(n)}$ (siendo $p' = kp$ con k constante), mientras que el número de funciones posibles está en $\mathcal{O}(2^{2^n})$. Así, de manera natural, surge preguntarse qué funciones no son computables con circuitos con un número de puertas polinómico.

La respuesta a esta pregunta es una gran incógnita en el campo de la informática teórica. Expongamos brevemente las implicaciones que dicha respuesta tendría sobre el resto de campos de la informática, y por extensión, sobre nuestra vida diaria. Denotamos por P a la clase de complejidad formada por el conjunto de problemas de decisión que pueden ser resueltos en tiempo polinómico por una máquina de Turing determinista y denotamos por NP al conjunto de problemas de decisión que pueden ser resueltos en tiempo polinómico por una máquina de Turing *no* determinista. Así,

el famoso problema P *vs* NP se pregunta si $P \subsetneq NP$. Este problema es famoso por su dificultad (no en vano es uno de los *problemas del millón de dólares* [3]) y por la revolución científica que conllevaría $P = NP$.

¿Qué tiene que ver pues P *vs* NP con los circuitos booleanos? En primer lugar, notemos que se define la clase de complejidad *P/poly* como el conjunto de lenguajes computables por una familia de circuitos booleanos de tamaño polinómico, uno por cada tamaño de entrada. Así, un resultado conocido [2] es que $P \subsetneq P/Poly$. Por tanto, cualquier problema que no pueda resolverse con una familia de circuitos de tamaño polinómicos no podrá estar en P. Es por ello que entender qué problemas requieren familias de circuitos de gran tamaño para ser computados abriría una nueva vía para abordar distintos problemas abiertos en el ámbito de la complejidad; como el ya mencionado P *vs* NP.

Sin embargo, nuestra meta será mucho menos ambiciosa, pues nos limitaremos a buscar qué factores influyen en el tamaño de los circuitos que computan una función booleana. Para ello, el enfoque que hemos tomado es el siguiente: en un circuito booleano irredundante, es decir, en el que existe un camino entre la puerta de salida y todas las demás puertas, si la anchura del mismo está limitada, en general habrá un nivel en el que las puertas se reutilizan, es decir, hay dos puertas que tienen como uno de sus inputs una misma tercera puerta. A esta situación la denominamos *endogámica*, ya que hay una fuerte dependencia entre las puertas de un nivel y las de su nivel inferior. Así, parece razonable preguntarse si esta reutilización reiterada de puertas coarta la expresividad del circuito y genera, en consecuencia, funciones calculadas por otros circuitos más pequeños. Haciendo una analogía con el caso del árbol genealógico, la endogamia en un circuito booleano sería equiparable a la endogamia en una familia en la que, por ejemplo, los padres de un hijo sean hermanos, primos, primos segundos... Desgraciadamente, las causas y los efectos de estas relaciones nos son desconocidos, por lo que navegar en el mar de la endogamia es un viaje a ciegas que quizás aporte resultados interesantes o quizás encallemos a medio camino.

Por ello, se ha diseñado un experimento para tratar de medir este fenómeno. En primer lugar, se generan de manera exhaustiva circuitos con profundidad y anchura fijas, incrementándose ambas paulatinamente. Así, computando la función booleana asociada al mismo, podemos guardar en una tabla indexada por la función computada el menor circuito computado (por ejemplo, menor número de puertas, menor profundidad...). Aún a pesar de la dispersión que de dicha tabla se espera, por los resultados del libro *Computational Complexity - A Modern Approach* mencionados anteriormente, elegir tamaños de input excesivamente grandes convierten en un cuello de botella la representación de las funciones computadas. Por ello, se ha optado por tomar 5 bits de input, de manera que el número máximo de funciones obtenibles con este experimento es de $2^{2^5} = 2^{32}$, que coincide con el máximo número de enteros representables en la mayoría de lenguajes de programación.

Tras la ejecución del mismo, tenemos por una parte información del tamaño y estructura de un circuito que computa dicha función, así como información de aquellas funciones que no aparezcan en dicha tabla: requieren circuitos con profundidad o anchura mayores de lo considerado. Por ello, la información obtenida del mismo no es baladí: puede servirnos para obtener evidencia empírica del fenómeno de la endogamia. Esta evidencia trataremos de ponerla de relieve mediante el análisis de distintas variables aleatorias y la correlación existente entre las mismas.

Además, otro fenómeno relacionado con las funciones booleanas es el de la *repetitividad* de patrones en la secuencia de bits que definen esta función. Este fenómeno, más etéreo, consiste en la existencia de una relación indeterminada entre la complejidad en el patrón de una función booleana

y el circuito que computa dicha función. Es etéreo en tanto que su existencia no tiene una base razonable más allá de la intuición de que a mayor complejidad del patrón de una función se espera mayor número de puertas del menor circuito que la compute (donde menor en este contexto se refiere al número de puertas empleado). Analizar este fenómeno es equivalente a analizar relaciones entre gramáticas que definan funciones y circuitos booleanos y esto se discutirá a lo largo del documento.

En resumen, a lo largo de este trabajo definiremos matemáticamente los conceptos de circuito booleano de profundidad n y estudiaremos algunas de sus propiedades más interesantes como la cardinalidad del conjunto que contiene a todos estos circuitos o la posibilidad de dotar al conjunto de un orden total. Además, proporcionaremos un algoritmo que permita recorrer de manera eficiente dicho conjunto para poder obtener información sobre los circuitos encontrados; información que será procesada y analizada mediante diversas métricas novedosas de endogamia definidas en este trabajo en aras de obtener algún tipo de relación entre las funciones booleanas y la endogamia latente en los circuitos que las computan.

No obstante, queda advertido el lector desde este primer instante de que todos los experimentos realizados lo son para una profundidad y anchura máximas fijadas de los circuitos booleanos, y en consecuencia, todas las conclusiones que se extraigan no son ni pretenden ser extrapolables de manera inmediata para cualquier profundidad. De hecho, debido a la explosión combinatoria de circuitos posibles, ni siquiera son un resultado en piedra para dichos valores seleccionados: el número de circuitos analizado es considerablemente escaso para la inmensidad de dicho espacio.

Aún así, aunque este documento no proporcione unas conclusiones teóricas trascendentales fruto del experimento, los resultados obtenidos pueden servir de guía o de pilar básico para una posterior investigación en este ámbito.

Plan de trabajo

Para realizar este trabajo se realizó la siguiente planificación temporal:

- Lectura en profundidad del capítulo 6 del libro *Computational Complexity - A Modern Approach* [2] (septiembre - octubre 2019).
- Desarrollo de un algoritmo eficiente para recorrer el espacio de circuitos (octubre 2019 - enero 2020).
- Paralelización del algoritmo (enero - febrero 2020).
- Desarrollo de distintas métricas de endogamia, estudio teórico del espacio de circuitos y su relación con el espacio de funciones booleanas e inicio de experimentos (marzo 2020).
- Definición de la repetitividad de una función booleana y estudio computacional del coste de las métricas de endogamia (abril - mayo 2020).
- Análisis de los resultados (mayo 2020).
- Desarrollo de la memoria final (mayo - junio 2020).

Sin embargo, dicha planificación se vio totalmente desbordada por la naturaleza del algoritmo, con lo que a la postre el desarrollo temporal de este experimento ha sido el siguiente:

- Lectura en profundidad del capítulo 6 del libro *Computational Complexity - A Modern Approach* [2] (septiembre - octubre 2019).
- Desarrollo de un algoritmo eficiente para recorrer el espacio de circuitos (octubre 2019 - mayo 2020). Una primera versión para determinadas profundidades y anchuras en función del tamaño del input estuvo terminada a principios de abril de 2020.
- Paralelización del algoritmo (enero - abril 2020).
- Desarrollo de distintas métricas de endogamia, estudio teórico del espacio de circuitos y su relación con el espacio de funciones booleanas e inicio de experimentos (abril 2020).
- Definición de la repetitividad de una función booleana y estudio computacional del coste de las métricas de endogamia (abril - mayo 2020).
- Análisis de los resultados (junio 2020).
- Desarrollo de la memoria final (mayo - junio 2020).

Capítulo 1

El espacio de circuitos

En este capítulo definiremos formalmente los circuitos booleanos y analizaremos diversas propiedades existentes en el espacio generado por dichos circuitos, como por ejemplo su cardinalidad o un orden total que nace en él de manera natural. Además, lo haremos con la suficiente abstracción y formalidad de manera que los resultados obtenidos en este espacio sean fácilmente trasladables a otros contextos.

1.1. Definiciones previas

En primer lugar, tengamos en cuenta que toda función booleana en n-bits, está unívocamente determinada por el valor 0 o 1 que toma para cada una de las 2^{2^n} combinaciones que pueden tomar dichos valores. Por tanto, es equiparable con una tupla de ${\mathbb{F}_2}^{2^n}$. Analicemos en consecuencia brevemente algunos conceptos clave de las funciones booleanas.

Definición 1.1.1. Sea $f \in \mathbb{F}_2[x_1, ..., x_n]$ un polinomio sobre $\mathbb{F}_2$. Llamamos *función evaluación*, a la función $ev \in \mathbb{F}_2[x_1, ..., x_n] \to {\mathbb{F}_2}^{2^n}$ de manera que $\forall k, 0 \leq k \leq 2^n - 1, (\pi_k \circ ev)(f) = f(bin_n(k))$, donde $bin_n(k)$ es la expresión en binario del número k usando n bits y π_k es la función proyección sobre la componente k-ésima.

Téngase en cuenta que dados n-bits de entrada, tenemos 2^n posibles maneras de elegir los valores que tome dicha función, por ello la imagen de ev está en ${\mathbb{F}_2}^{2^n}$. Luego es claro que podemos establecer una relación entre las funciones booleanas y el conjunto de polinomios que computan dicha función.

Definición 1.1.2. Sea $\mathcal{C}$ un conjunto arbitrario. Denominamos *operador n-polinomizador* a cualquier función $f \in \mathcal{C} \to \mathbb{F}_2[x_1, ..., x_n]$, donde $\mathcal{C}$ es un conjunto arbitrario.

La función de este operador consiste en explicitar la diferencia entre un circuito y la función booleana que este computa. Así, como veremos ahora con la ayuda de estos conceptos, podemos establecer una relación entre ambos conceptos mediante un diagrama conmutativo.

Definición 1.1.3. Llamaremos *operador evaluación* a cualquier función $o_e : \mathcal{C} \to \mathbb{F}_2{}^{2^n}$. Usualmente emplearemos el operador evaluación $ev_\#$ definido así: $ev_\# = ev \circ pol$, donde pol es un operador n-polinomizador.

Es decir, el operador n-polinomizador es una herramienta que nos permite asociar elementos de un conjunto $\mathcal{C}$ a polinomios sobre $\mathbb{F}_2$ y en consecuencia, poder hablar de la evaluación de los mismos.

Definición 1.1.4. Sean $\mathcal{C}, \mathcal{C}'$ conjuntos. Llamaremos a un operador $P \in \mathcal{C} \times \mathcal{C} \to \mathcal{C}'$ *puerta lógica* si existe un operador $eq(P) \in \mathbb{F}_2{}^{2^n} \times \mathbb{F}_2{}^{2^n} \to \mathbb{F}_2{}^{2^m}$ denominado *equivalente* tal que $\forall\ C_1,\ C_2 \in \mathcal{C}$ cumple que $(oe_m \circ P)(C_1, C_2) = eq(P)(oe_n(C_1),\ oe_n(C_2))$.

Definición 1.1.5. Diremos que un operador $P : \mathcal{C} \times \mathcal{C} \to \mathcal{C}'$ bidimensional es *simétrico* si $\forall C_1, C_2 \in \mathcal{C}, (oe \circ P)(C_1, C_2) = (oe \circ P)(C_2, C_1)$.

Aunque habitualmente emplearemos $n = m$ y omitiremos el subíndice del operador oe, esta definición nos permite interaccionar con distintos conjuntos en los que cada uno se emplee un operador evaluación distinto. Así, combinando esta con la anterior observación, observamos que en la relación anterior podemos escribir lo siguiente: $oe_n = oe_m = ev_\#$. Además, a partir de ahora supondremos que tenemos un conjunto $\mathcal{P}$ de puertas que estará de manera implícita en todas las definiciones aunque no se hable del mismo por claridad en la notación. Por simplicidad, supondremos que todas las puertas lógicas de $\mathcal{P}$ son simétricas. Con todos estos matices, podemos continuar abstrayendo nuevamente la idea de circuito.

Definición 1.1.6. Denotamos por $\mathcal{I}_m$ a un conjunto de cardinalidad m y a cada uno de los elementos de $\mathcal{I}_m$ lo denominamos como *input* y lo escribiremos como $in_i, 0 \leq i < m$.

A partir de ahora y mientras no se diga lo contrario, supongamos el cardinal de $\mathcal{I}_m$ fijo de manera que todas las definiciones que prosiguen a esta frase están de manera implícita sujetas a dicho valor. Notemos que podemos establecer una equivalencia entre el input i-ésimo, in_i, y el polinomio x_i de $\mathbb{F}_2[x_0, ..., x_{m-1}]$.

Definición 1.1.7. Llamamos *etiqueta* a un elemento de un conjunto E arbitrario y llamamos *mapa de etiquetas* a una función $f : E \to \mathcal{C}'$, donde E y $\mathcal{C}'$ son dos conjuntos arbitrarios.

Por simplicidad, nosotros trabajaremos con $E = \mathbb{Z}$.

Definición 1.1.8. Denominamos por *circuito de profundidad 0* a todo C perteneciente a $\mathcal{I}_m$ y al conjunto de estos elementos lo llamaremos $\mathcal{C}_0$. Es decir, $\mathcal{I}_m = \mathcal{C}_0$.

Con esta definición, es claro que hay m circuitos de profundidad 0, a saber, los elementos $in_i \in \mathcal{I}_m$.

Definición 1.1.9. Dado $n > 0$ y un conjunto de mapa de etiquetas $F = \{f_i,\ 1 \leq i \leq n,\ f_n : E \to \mathcal{P} \times \mathcal{C}_{n-1} \times \mathcal{C}_{n-1}\}$, denominamos *circuito de profundidad n*, a la tupla (C, P, C_1, C_2) donde P es una puerta lógica, C_1, C_2 son dos etiquetas asociadas a dos circuitos de profundidad $n-1$ y C es una etiqueta tal que $f_n(C) = (P, C_1, C_2)$ y lo denotaremos por $C \coloneqq P(C_1, C_2)$. Al conjunto de estos elementos lo denotaremos por $\mathcal{C}_n$.

Definición 1.1.10. Dada una colección de mapas de etiquetas $F = \{f_i, i \in \mathbb{N}\}$ denotamos con $\mathcal{C} = \bigcup_{n\in\mathbb{N}} \mathcal{C}_n$.

Con esta definición de circuito queremos expresar formalmente lo que intuitivamente conocemos como circuito lógico. Así, cuando denotamos $C \coloneqq P(C_1, C_2)$ expresamos la idea de conectar a la puerta lógica física dos entradas provenientes de los circuitos C_1 y C_2, de manera que en esta definición eliminamos cualquier tipo de bucle y se puede ver como un grafo dirigido con raíz.

Notemos además que no hemos definido como circuito ni $\top$ ni $\bot$, es decir, el circuito físico conectado a voltaje 1 y 0. Esto es así ya que resultan inconveniente en las definiciones y no aporta demasiado contar dos circuitos cuya evaluación se puede obtener con circuitos de profundidad 1.

Finalmente destaquemos que no hemos especificado en ningún momento los mapas de etiquetas empleados, ni tampoco nos interesa hacerlo: el objetivo final es prescindir de ellos eliminando las repeticiones en $\mathcal{C}$ que estudiaremos más adelante, de manera que cada circuito llegue a verse de manera única y podamos además establecer un mapa de etiquetas "canónico". Por ello, todo lo que se diga a partir de ahora es sujeto al conjunto de mapas F que tomemos.

La notación $C \coloneqq P(C_1, C_2)$ es una notación ecuacional de los circuitos y será muy útil para establecer relaciones de equivalencia. Al circuito C lo llamaremos *miembro izquierdo* y a los circuitos C_1 y C_2 *miembros derechos*. Notemos además que cada circuito se define de una única manera. Esta observación motiva la siguiente definición.

Definición 1.1.11. Denotaremos con $E(C)$ al conjunto mínimo de ecuaciones que describen un circuito C.

Dado un circuito $C \coloneqq P(C_1, C_2)$, es claro que para definir C necesitamos saber la definición de C_1 y C_2. Así, podemos hallar de manera recursiva todas las ecuaciones que definen C de manera que tengamos su descripción de manera exacta y estas y solo estas serán las que formarán $E(C)$. Además y por inercia del lenguaje natural, diremos que el circuito C tiene $|E(C)|$ puertas lógicas. Así, teniendo en mente la imagen usual de un circuito booleano, $E(C)$ representa a todas las puertas lógicas empleadas.

Definición 1.1.12. Dado un circuito C de profundidad n, un circuito C' de profundidad $k, 0 \leq k \leq n$ es un *subcircuito* de C, y lo denotaremos $C' \sqsubseteq C$, si $E(C') \subseteq E(C)$.

Definición 1.1.13. Dado $l \in \mathbb{N}$ denotaremos por $C' \sqsubseteq_l C$ a todo par de circuitos C, C' tales que $C' \sqsubseteq C$ y C' tiene profundidad l. Si $C' \neq C$, entonces podremos denotarlo $C' \sqsubset_l C$. Además, denotaremos por $E_l(C)$ al conjunto de ecuaciones del tipo $C' \coloneqq P(C_1, C_2) \in E(C)$ tales que C' tenga profundidad l.

Además, denotaremos por $|E_0(C)|$ al número de elementos de $\mathcal{I}_m$ que aparecen en dichas ecuaciones. Es decir, extenderemos la notación de manera natural teniendo en cuenta que $|E(C)| = \sum_{l=1}^{n} |E_l(C)| \neq \sum_{l=0}^{n} |E_l(C)|$. Esto se debe a que intuitivamente $|E(C)|$ representa el número de puertas de C y los circuitos de profundidad 0 no tienen puertas. Sin embargo, podemos ver $|E_0(C)|$ como el número de inputs que el circuito C utiliza. Este abuso de notación será muy útil posteriormente.

Lema 1.1.14. *Todo circuito de profundidad n tiene al menos n puertas.*

Demostración. Los circuitos de profundidad 0 tienen 0 puertas. Además, todo circuito de profundidad n utiliza al menos una puerta combinando al menos un circuito de profundidad $n-1$. Por tanto, la afirmación es inmediata. □

Corolario 1.1.15. *La cota del lema 1.1.14 se alcanza.*

Demostración. Sea el circuito definido recursivamente como:

$$X_{n,k} \coloneqq \begin{cases} id(X_{n-1,k}, X_{n-1,k}) & \text{si} \quad n > 0 \\ x_k & \text{si} \quad n = 0 \end{cases} \tag{1.1}$$

donde x_k es el k-ésimo input de $\mathcal{I}_m$. De manera trivial este circuito utiliza exactamente n puertas. □

Lema 1.1.16. *Todo circuito de profundidad n tiene a lo sumo $2^n - 1$ puertas.*

Demostración. Demostrémoslo por inducción. Los circuitos de profundidad 0 tienen 0 puertas. Así, supongámoslo probado para $n-1$ y estudiémoslo para $n > 0$: un circuito de profundidad n es de la forma $C \coloneqq P(C_1, C_2)$. Entonces $|E(C)| \leq 1 + |E(C_1)| + |E(C_2)| \leq 1 + 2^{n-1} - 1 + 2^{n-1} - 1 = 2^n - 1$. □

Ya que en el conjunto $\mathcal{I}_m$ hay exactamente m elementos, diremos indistintamente que los circuitos de profundidad 0 son iguales o idénticos, significando en este caso lo mismo y lo denotaremos por $C = C'$ o por $C \equiv C'$ respectivamente.

Definición 1.1.17. Diremos que dos circuitos de profundidad $n > 1$, $C \coloneqq P(C_1, C_2)$ y $C' \coloneqq P'(C_1', C_2')$ son *idénticos* si se da que, o bien $C_1 \equiv C_1'$, $C_2 \equiv C_2'$ y $P = P'$, o bien $C_1 \equiv C_2'$, $C_2 \equiv C_1'$ y $P = P'$; y lo denotaremos $C \equiv C'$.

Del hecho de que los inputs y las puertas se preserven entre circuitos idénticos, es claro que para cualesquiera dos circuitos idénticos, C y C', tanto en C como en C' hay el mismo número de puertas lógicas en cada nivel, y hay una biyección entre las ecuaciones de cada uno de los circuitos. Por ello, surge la siguiente definición.

Definición 1.1.18. Dados dos circuitos C, C' idénticos, a la biyección anterior entre $E(C)$ y $E(C')$ que preserva puertas e inputs la denominaremos *renombramiento* de C en C'.

Comparemos esta definición con la de igualdad para circuitos de profundidad $n > 1$. Así, si $C \coloneqq P(C_1, C_2)$, $C' \coloneqq P(C_1, C_2)$, es claro que $C \equiv C'$ pero no podríamos decir que $C = C'$. Esto intuitivamente quiere decir que si partimos de dos circuitos C_1 y C_2 y construimos físicamente dos puertas lógicas uniendo en cada una mediante dos cables la puerta con C_1 y C_2, los circuitos resultantes serían equivalentes a nivel estructural, es decir, serían idénticos; pero no serían iguales, ya que físicamente utilizan dos puertas lógicas distintas y en consecuencia, como circuitos de puertas lógicas, son distintos.

Así, solo diremos que dos circuitos lógicos son iguales cuando la etiqueta con la que nos refiramos a ellos sea la misma. En otras palabras quizás más visuales, solo diremos que son idénticos cuando la puerta lógica física que sustenta esta idea sea exactamente la misma.

Además, hagamos notar lo siguiente: al ser las puertas lógicas operadores simétricos, las tuplas (P, C_1, C_2) y (P, C_2, C_1) se comportan igual ante la evaluación y ante identidad y no tiene mucho sentido considerarlos distintos ya que en la idea intuitiva que hay por detrás de circuito físico se da que son exactamente iguales. Por ello, y durante del resto del documento, trabajaremos módulo simetría de los operadores. Es decir, se hablará indistintamente del circuito $C \coloneqq P(C_1, C_2)$ o de $C \coloneqq P(C_2, C_1)$ pero no se dirá que $C \coloneqq P(C_1, C_2)$ sea igual que $C' \coloneqq P(C_1, C_2)$.

Lema 1.1.19. *Si C_1 y C_2 son dos circuitos idénticos, $ev_\#(C_1) = ev_\#(C_2)$.*

Demostración. Razonemos por inducción. Si C_1 y C_2 son dos circuitos de profundidad 0, entonces, por definición, son iguales. Luego solo puede suceder que $ev_\#(C_1) = ev_\#(C_2)$. Supongamos que para cualesquiera dos circuitos de profundidad n se cumple el lema y sea $C = P(C_1, C_2)$, $C' = P'(C_1', C_2')$. Como $C \equiv C'$, entonces $P = P'$ y, o bien $C_1 \equiv C_1', C_2 \equiv C_2'$, o bien $C_1 \equiv C_2', C_2 \equiv C_1'$. Por tanto $ev_\#(C) = (ev_\# \circ P)(C_1, C_2) = eq(P)(ev_\#(C_1), ev_\#(C_2)) = eq(P)(ev_\#(C_1'), ev_\#(C_2')) = (ev_\# \circ P)(C_1', C_2') = ev_\#(C')$, donde se ha aplicado la hipótesis de inducción junto a la simetría de la puerta para garantizar que $eq(P)(ev_\#(C_1), ev_\#(C_2)) = eq(P)(ev_\#(C_1'), ev_\#(C_2'))$. □

A partir de ahora consideremos las puertas lógicas pertenecientes al conjunto $\mathcal{P} = \{\wedge, \vee, id\}$ definidas así:

$$\begin{array}{lll} C \coloneqq \wedge(C_1, C_2) & \text{si} & C_1 \neq C_2 \\ C \coloneqq \vee(C_1, C_2) & \text{si} & C_1 \neq C_2 \\ C \coloneqq id(C_1, C_2) & \text{si} & C_1 = C_2 \end{array}$$

donde C_1 y C_2 son dos circuitos de la misma profundidad, la igualdad no es vía renombramiento y dichas puertas tienen un equivalente natural, el *and* y el *or* lógico de n variables y la función identidad respectivamente.

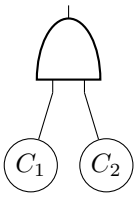


Figura 1.1: Representación física del circuito $C \coloneqq \wedge(C_1, C_2)$.

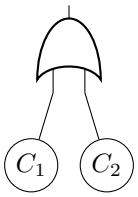


Figura 1.2: Representación física del circuito $C \coloneqq \vee(C_1, C_2)$.

Figura 1.3: Representación física del circuito $C \coloneqq id(C_1, C_1)$.

Para poder hablar de la evaluación de los circuitos con las puertas elegidas, necesitamos dotar a $\mathcal{C}$ de un m-polinomizador, donde m es el número de inputs. Aunque hay infinitas aplicaciones de esta naturaleza, mostraremos a continuación una manera natural de hacerlo.

Lema 1.1.20. *El espacio $\mathcal{C}$ tiene un m-polinomizador compatible con la biyección entre polinomios x_i y circuitos de profundidad* 0*, al cual llamaremos* m-polinomizador usual*.*

Demostración. Si C es un circuito de profundidad 0, es equivalente a un polinomio x_i, con lo que de manera natural asociaremos C con x_i. Además, a circuitos idénticos asocia polinomios iguales,

lo cual es importante. Supongamos pues que tenemos el m-polinomizador usual definido para todo circuito de profundidad n de manera que circuitos idénticos tengan misma imagen y tomemos un circuito $C \coloneqq P(C_1, C_2)$ de profundidad $n+1$. Entonces tenemos tres posibilidades:

- $\underline{P = id}$. En este caso diremos que $f(C') = f(C_1)^2$, ya que $f(C_1) = f(C_2)$.
- $\underline{P = \vee}$. En este caso definiremos $f(C') = f(C_1) + f(C_2)$.
- $\underline{P = \vee}$. Así, tomaremos $f(C') = f(C_1) \cdot f(C_2)$.

Finalmente, es inmediato comprobar que circuitos idénticos tienen misma imagen gracias a la conmutatividad del anillo $\mathbb{F}_2[x_1, ..., x_m]$.

□

Corolario 1.1.21. *Todo circuito de profundidad n tiene asociado un polinomio de multigrado n.*

Demostración. Es inmediato debido a la construcción del m-polinomizador usual. □

Lema 1.1.22. *Sean C_1, C_2 y C_3 tres circuitos de profundidad n tales que $C_1 \equiv C_2$ y $C_1 \neq C_2$. Entonces $(ev_\# \circ \wedge)(C_1, C_2) = (ev_\# \circ \vee)(C_1, C_2) = (ev_\# \circ id)(C_1, C_1)$ y $(ev_\# \circ P)(C_1, C_3) = (ev_\# \circ P)(C_2, C_3)$, donde P es una puerta lógica.*

Demostración. Si $C_1 \equiv C_2$ entonces se da que $ev_\#(C_1) = ev_\#(C_2)$ por el lema 1.1.19. Así, utilizando las propiedades del equivalente, es claro que $eq(\wedge)(ev_\#(C_1), ev_\#(C_2)) = eq(id)(ev_\#(C_1), ev_\#(C_1)) = eq(\vee)(ev_\#(C_1), ev_\#(C_2))$; ya que el or y el and lógicos aplicados a la misma tupla se comportan como la identidad. De este modo, la primera parte del lema queda probada. Finalmente, es inmediato concluir la última parte puesto que $(ev_\# \circ P)(C_1, C_3) = eq(P)(ev_\#(C_1), ev_\#(C_3)) = eq(P)(ev_\#(C_2), ev_\#(C_3)) = (ev_\# \circ P)(C_1, C_3)$. □

Este lema junto al 1.1.19 nos hace ver que los circuitos idénticos, aun no siendo iguales, son intercambiables gracias siempre a los equivalentes naturales que hemos elegido. Esto propiciará más adelante que tratemos de evitar repeticiones dentro de las ecuaciones de un circuito, las cuales serán generadas debido a la presencia de dichos circuitos idénticos.

Por otra parte, al lector avezado no se le habrá escapado la ausencia en esta elección de puertas lógicas negadoras. Esto se debe al siguiente resultado:

Teorema 1.1.23. *Todo circuito lógico generado por las puertas $\wedge, \vee, id$ y $\neg$ tiene un circuito lógico equivalente donde todos los $\neg$ se encuentren en el nivel de profundidad 0.*

Demostración. En primer lugar, observemos que todos los $\neg$ se pueden ver como un añadido a las puertas $\wedge, \vee$ e id, es decir, $\neg \circ \wedge$, $\neg \circ \vee$ y $\neg \circ id$. Así, dado un circuito arbitrario, podemos considerar que los $\neg$ solo siguen a una puerta lógica de la manera explicada.

Razonemos una vez más por inducción. Si el circuito lógico tiene profundidad 0, es inmediato. Así, supongámoslo cierto para el nivel n y tomemos un circuito de profundidad $n+1$ definido así: $C' \coloneqq P(C_1, C_2)$. Si $P \in \{\wedge, \vee, id\}$, ya habríamos acabado.

Por ello, supongamos que $P \in \{\neg \circ \wedge, \neg \circ \vee, \neg \circ\ id\}$. En ese caso, podemos definir un circuito $C' \coloneqq P'(\neg(C_1), \neg(C_2))$, donde $P' = Q$ si $P = \neg \circ\ Q$; el cual es equivalente a C. Por hipótesis de inducción, existe un circuito C'' equivalente a C' donde todos los $\neg$ están en el nivel 0. Por tanto, al ser C equivalente a C', queda acabada la prueba. □

La utilidad de este teorema radica en lo siguiente: dado un circuito arbitrario C con k inputs generado por $\wedge, \vee, id$, lo podemos representar como un circuito con $m = 2k$ inputs, donde los primeros k inputs los identificaremos con $(x_0, ..., x_{k-1})$ y los k siguientes con $(1 - x_0, ..., 1 - x_{k-1})$ (equivalentes a $\neg(x_i)$). Por ello, podemos olvidarnos de las puertas $\neg$ y suponer de ahora en adelante que el tamaño del input viene dado por $m = 2k$. Sin embargo, los resultados que se expondrán no lo tendrán en cuenta, es decir, solo influirán estas relaciones a la hora de evaluar el circuito, no de construirlo.

1.2. Representación minimal y maximal

En esta sección mostraremos cómo transformar un circuito C en otro C' de misma profundidad apoyándonos en la existencia de circuitos idénticos.

Algoritmo 1.2.1 (Transformación minimal)**.** *Dado un circuito C de profundidad n devuelve un circuito C' de la misma profundidad que no tiene dos subcircuitos idénticos. Además, C' es único salvo renombramiento.*

Demostración. La idea de este algoritmo será ir construyendo una colección de circuitos $C_i, 0 \leq i < n$ identificando circuitos idénticos, y esto lo haremos por inducción cambiando el mapa f_{i+1} en cada paso si fuese necesario. Todo circuito C no tiene circuitos de profundidad 0 idénticos por construcción, con lo que tomemos $C_0 = C$. Así, supongamos que tenemos definido C_{k-1} y que en $E(C_{k-1})$ no hay dos circuitos de profundidad $j, 0 \leq j \leq k-1$ idénticos.

Construyamos pues un circuito C_k sin circuitos idénticos de profundidad $j,\ 0 \leq j \leq k < n$. Si no tiene dos subcircuitos de profundidad k idénticos, hemos acabado y $C_{k+1} = C_k$. En caso contrario, sean $D, D' \sqsubset_k C_{k-1}$ de manera que $D \equiv D'$. Entonces, para todo circuito de la forma $E \coloneqq P'(D', \hat{D}')$, con $D' \not\equiv \hat{D}', D' \equiv D$ redefinamos $f_{k+1}(E) = (P', D, \hat{D}')$. Por otra parte, para todo circuito $E \coloneqq P'(D'\hat{D}')$ con $D' \equiv \hat{D}' \equiv D$, redefinamos $f_{k+1}(E) = (id, D, D)$. Finalmente denotemos por C'_k al circuito de profundidad n con etiqueta C definido con el nuevo mapa.

El circuito C'_{k-1} obtenido con esta nueva colección de mapas cumple que tiene un circuito de profundidad k menos y por tanto, menos pares de circuitos de profundidad k idénticos entre sí. Si este proceso lo repetimos para todos los circuitos de profundidad k de manera iterada obtenemos

un mapa modificado, f_{k+1}, de manera que en el circuito resultante no hay subcircuitos idénticos de profundidad k. Así, tenemos perfectamente definido C_k y en consecuencia, por inducción, $C_n = C'$.

Finalmente el además es inmediato a la vista de la demostración: otra elección del circuito D a la hora de modificar el mapa f_k nos conduce a un circuito C'' que es un renombramiento de C'. □

Definición 1.2.2. Diremos que un circuito C está en su *representación minimal* si es idéntico al circuito C' obtenido al aplicar el algoritmo 1.2.1 al circuito C.

Intuitivamente, la representación minimal de un circuito consiste en expresar dicho circuito con el menor número de puertas manteniendo la misma estructura, eliminando puertas repetidas. Sin embargo, destaquemos que esto no significa que el circuito que computa $f(C)$ sea mínimo, donde f es el operador m-polinomizador usual. A mayores, destaquemos que esta representación minimal es única ya que el algoritmo anterior es determinista.

Corolario 1.2.3. *Para todo circuito C y su representación minimal C' se da que $ev_{\#}(C) = ev_{\#}(C')$.*

Demostración. Es inmediato siguiendo la demostración del algoritmo 1.2.1 y aplicando sucesivamente los lemas 1.1.19 y 1.1.22. □

Sin embargo, no es cierto que un circuito y la representación minimal de dicho circuito sean idénticos, como se puede ver en el siguiente ejemplo.

Ejemplo 1.2.4. Sean los circuitos C y C' dados por las siguientes ecuaciones:

$$E(C) = \begin{cases} C \coloneqq \wedge(D_1, D_2) \\ D_1 \coloneqq id(in_0, in_0) \quad D_2 \coloneqq id(in_0, in_0) \end{cases} \tag{1.2}$$

$$E(C') = \begin{cases} C' \coloneqq id(D', D') \\ D' \coloneqq id(in_0, in_0) \end{cases} \tag{1.3}$$

Es claro que C' es la representación minimal de C pero $C \not\equiv C'$ ya que $C \coloneqq \wedge(D_1, D_2)$, $C' \coloneqq id(D', D')$ y $\wedge \neq id$. Intuitivamente esto es razonable ya que no es lo mismo crear un circuito usando dos puertas lógicas que usando tres.

Pero sin embargo, es claro que son *esencialmente* el mismo circuito eliminando puertas redundantes, por lo que ampliaremos la definición de circuitos idénticos diciendo que dos circuitos son idénticos si sus representaciones minimales son idénticas.

Algoritmo 1.2.5 (Transformación maximal)**.** *Dado un circuito C de profundidad n devuelve un circuito C' de la misma profundidad con $2^n - 1$ puertas.*

Demostración. La idea de este algoritmo será ir construyendo una colección de circuitos $C_i, p \leq i \leq 2^n - 1$, con $p = |E(C)|$, de manera que se dividan duplicidades para que florezcan circuitos idénticos y en cada paso aparezca una nueva puerta más; y esto lo haremos por inducción cambiando quizás

algún mapa de etiquetas en cada paso si fuese necesario. Dado que $p = |E(C)|$, tomemos $C_p = C$. Así, supongamos que tenemos definido $C_{k-1},\ p \leq k-1 < n$ y que $|E(C_{k-1})| = k-1$ y generemos C_k.

Si $k-1 < n$ implica que hay al menos una etiqueta D utilizada dos veces asociada a un circuito de profundidad $l > 0$. Sea además D' una etiqueta no utilizada por ningún subcircuito de profundidad l de C. Así, si existe un circuito de profundidad $l+1$ definido como $E \coloneqq id(D, D)$, y ampliemos la definición de f_l diciendo $f_l(D') = f_l(D)$ y modifiquemos la definición de $f_{l+1}(E) = (P, D, D')$, donde $P \in \{\wedge, \vee\}$ y esta puerta se elige de manera arbitraria. Si no, es porque existen dos circuitos $E, E' \sqsubseteq_{l+1} C$ definidos así: $E \coloneqq P(D, C_1), E' \coloneqq P'(D, C_2)$, donde necesariamente $P, P' \in \{\wedge, \vee\}$. En este caso definamos $f_l(D') = f_l(D)$ y modifiquemos las definiciones de E, E' así: $f_{l+1}(E) = (P, D, C_1), f_{l+1}(E') = (P', D', C_2)$.

En cualquiera de los dos casos el número de puertas ha aumentado en 1, con lo que podemos denotar por C_k al circuito de profundidad n con etiqueta C definido con el nuevo mapa. En este caso se da que $|E(C_k)| = k$, con lo que en un número finito de pasos hemos acabado. □

Definición 1.2.6. Diremos que un circuito C de profundidad n está en una *representación maximal* si es idéntico a alguno de los circuitos C' posibles que se pueden obtener a partir de C con el algoritmo 1.2.5.

Corolario 1.2.7. *La representación maximal de un circuito puede no ser única.*

Demostración. Se sigue del hecho de que en el algoritmo 1.2.5 si existe un circuito de la forma $E \coloneqq id(D, D)$ lo podemos sustituir por $E \coloneqq \wedge(D, D')$ o por $E \coloneqq \vee(D, D')$ de manera indistinta y los circuitos obtenidos son distintos (y no idénticos). □

Corolario 1.2.8. *Si la representación minimal de un circuito C de profundidad n tiene $2^n - 1$ puertas, la representación maximal es única y coincide con la representación minimal.*

Demostración. En primer lugar, por el lema 1.1.16 todo circuito tiene a lo sumo $2^n - 1$. Entonces, dado que la representación minimal es única y tiene $2^n - 1$ puertas, el algoritmo 1.2.5 devuelve un circuito idéntico a C. □

Corolario 1.2.9. *La representación maximal de un circuito C puede ser única y no coincidir con su representación minimal.*

Demostración. Sea el circuito C definido por las ecuaciones:

$$E(C) = \begin{cases} C \coloneqq \wedge(D_0, D_1) & & \\ D_0 \coloneqq \vee(E_0, E_2) & D_1 \coloneqq \wedge(E_1, E_2) & \\ E_0 \coloneqq id(in_0, in_0) & E_1 \coloneqq id(in_1, in_1) & E_2 \coloneqq id(in_2, in_2) \end{cases} \tag{1.4}$$

Es claro que no está en representación maximal ya que $|E(C)| = 6 \neq 7 = 2^3 - 1$. Esto se debe a que la etiqueta E_2 se utiliza dos veces en las definiciones. Sin embargo, su representación maximal es única ya que solo hay una manera de definir una etiqueta E_3 y modificar la definición de $f_2(D_1)$, a saber:

$$E(C') = \begin{cases} C' \coloneqq \wedge(D_0, D_1) & & & \\ D_0 \coloneqq \vee(E_0, E_2) & D_1 \coloneqq \wedge(E_1, E_3) & & \\ E_0 \coloneqq id(in_0, in_0) & E_1 \coloneqq id(in_1, in_1) & E_2 \coloneqq id(in_2, in_2) & E_3 \coloneqq id(in_2, in_2) \end{cases} \tag{1.5}$$

□

A partir de esta sección trabajaremos con circuitos en representación minimal, en los cuales se cumple que los términos identidad e igualdad son equivalentes. Todos los resultados los expondremos módulo identidad de circuitos, es decir, dos circuitos con la misma representación se considerarán el mismo. Además, los símbolos $\mathcal{C}_n$ y $\mathcal{C}$ se emplearán para el conjunto de circuitos en representación minimal de profundidad n y el conjunto de circuitos en representación minimal de profundidad arbitraria respectivamente.

1.3. Diversidad minimal

Analizaremos ahora la diversidad de circuitos de profundidad n que se pueden generar a partir de dos de profundidad $n-1$.

Definición 1.3.1. Denotamos por $\delta_m \in \mathbb{N} \to \mathbb{N}$ a la función *diversidad* definida por:

$$\delta_m(n) = m^{2^n} \tag{1.6}$$

Lema 1.3.2. *Si el número de circuitos de profundidad n es k, entonces el número de circuitos de profundidad $n+1$ es k^2.*

Demostración. Como el operador $\wedge$ solo se aplica sobre circuitos distintos de $\mathcal{C}_n$ y en $\mathcal{C}_n$ hay k circuitos, entonces el número de ternas $(\wedge, C_1, C_2)$, con $C_1 \neq C_2$ es $\binom{k}{2} = \frac{1}{2}k \cdot (k-1)$ (donde el factor $\frac{1}{2}$ aparece por la simetría del operador). De manera análoga, el número de ternas $(\vee, C_1, C_2)$ es $\frac{1}{2}k \cdot (k-1)$. Por último, como el operador identidad se aplica solo a circuitos idénticos, y hay k circuitos en $\mathcal{C}_n$, obtenemos que en $\mathcal{C}_{n+1}$ hay $2 \cdot \frac{1}{2}k \cdot (k-1) + k = k^2$. □

Corolario 1.3.3. *El número de circuitos de profundidad n generables a partir de m inputs es $\delta_m(n)$.*

Demostración. Si $n = 0$, los únicos circuitos que se pueden generar son todos aquellos que están en $\mathcal{I}_m$, y $|\mathcal{I}_m| = m = \delta_m(0)$.

Supongamos ahora que hemos probado el corolario para n, es decir, hay $\delta_m(n)$ circuitos en $\mathcal{C}_n$. Entonces por el lema 1.3.2 en $\mathcal{C}_{n+1}$ hay $\delta_m^2(n) = \delta_m(n+1)$ circuitos. □

Como se puede ver, el número de circuitos crece de manera doblemente exponencial con la profundidad. Este resultado es importante ya que nos dice que este espacio es inabarcable de manera práctica, solo para $m = 1$ es computable dicho espacio (el cual es trivial). Además, esto justifica el porqué a la hora de razonar sobre complejidad de problemas en el que intervengan circuitos se exija que la profundidad p de dichos circuitos venga dada por una función logarítmica ($p \in \mathcal{O}(\log(n))$), de manera que $|\mathcal{C}_p| = \delta_m(\log(n)) = m^n$. De esta manera, estaríamos trabajando sobre un espacio de tamaño exponencial, no doblemente exponencial, lo que puede aliviar la dificultad del problema con el que estemos trabajando.

1.4. Orden lexicográfico de circuitos

Finalizaremos este capítulo mencionando un orden total que podemos inducir en el espacio de circuitos $\mathcal{C}$. Este orden obtenido será la guía del algoritmo presentado en el capítulo 2, el cual se encargará de recorrer $\mathcal{C}$ aprovechando distintas propiedades de este orden lexicográfico.

En primer lugar, establezcamos un orden en el conjunto de inputs, esto es, digamos a partir de ahora que $x_0 < ... < x_{m-1}$. Así, hemos dotado de un orden a los circuitos de profundidad 0. Veamos que, solo del hecho de establecer un orden entre las puertas lógicas, se dota de manera natural de un orden a este conjunto.

Definición 1.4.1. Dado un circuito C de profundidad n y una función $f : \mathcal{C}_n \to \mathbb{Z}_{|\mathcal{C}_n|}$, denominamos *índice* de C vía f al valor $f(C)$. A dicha función f la denominaremos *función índice* y la denotaremos por f_n, donde n es la profundidad del circuito sobre el que se aplica.

Definición 1.4.2. Dadas dos puertas lógicas P, P' decimos que *P es menor que P'* si se da que para cualesquiera circuitos C_1, C_2 de profundidad n, $P(C_1, C_2) < P'(C_1, C_2)$. Así, dado un conjunto de puertas lógicas diremos que están *ordenadas* si dadas dos puertas P, P' o bien $P < P'$ o bien $P' < P$.

En nuestro caso podemos establecer que $\vee < \wedge$, pero no podemos comparar ninguna de ellas con la identidad. En cualquier caso, ya que podemos ver id como una extensión de $\wedge$, abusaremos de este concepto y diremos que existe un orden entre las puertas, eso es, que $\vee < \wedge \equiv id$. Esta decisión arbitraria, ya que podíamos haber visto id como una extensión del operador $\vee$, no tendrá a posteriori mucha influencia; simplemente se toma por el mero hecho de que queda extraño hablar de órdenes totales inducidos por conjuntos no comparables. Además, como se verá más adelante, solo usaremos este orden para desempatar entre circuitos distintos compuestos con los mismos argumentos (por lo que la decisión tomada sobre id no tendrá peso alguno).

De manera natural, es claro que cualquier función índice induce un orden sobre $\mathcal{C}_n$ con el orden natural de $\mathbb{Z}_{\delta_m(n)}$. A este orden inducido lo denominaremos *orden en el nivel n*. Sin embargo, estas funciones no solo inducen un orden sobre $\mathcal{C}_n$ sino que también sobre $\mathcal{C}_{n+1}$.

Lema 1.4.3. *Toda función índice f_n induce un orden sobre $\mathcal{C}_{n+1}$.*

Demostración. Sea f_n una función índice y sean $C \coloneqq P(C_1, C_2)$, $C' \coloneqq P'(C'_1, C'_2)$ dos circuitos de profundidad $n+1$ distintos. Entonces diremos que C es menor que C' vía f_n si $f_n(C_1) < f_n(C'_1)$,

o bien $f_n(C_1) = f_n(C_1')$ y $f_n(C_2) < f_n(C_2')$, o bien $f_n(C_1) = f_n(C_1'), f_n(C_2) = f_n(C_2'), P = \vee$ y $P' = \wedge$. Este orden es total en $\mathcal{C}_n$ ya que estamos trabajando sobre circuitos en representación minimal y por tanto identidad es sinónimo de igualdad. □

Sin embargo, este orden no dice nada sobre los circuitos de profundidad menor que n; y en consecuencia, no ordena internamente sus subcircuitos. Una idea naïf para extender esta definición podría ser intentar realizar un pequeño apaño al lema anterior en dirección contraria. Sin embargo, hacerlo así hace que al tomar dos circuitos de profundidad $n > 2$, el orden de los subcircuitos de profundidad $n-2$ no está unívocamente determinado, esto es, puede darse que $C \neq C'$, $E_{n-2}(C) = E_{n-2}(C')$ y el orden inducido sobre este subconjunto de circuitos de $\mathcal{C}_{n-2}$ sea distinto. Veámoslo con un ejemplo.

Ejemplo 1.4.4. Sean los circuitos C y C' de profundidad $n > 2$ que vienen dados por las ecuaciones:

$$E(C) = \begin{cases} & & C \coloneqq \wedge(C_1, C_2) & \\ & & C_1 \coloneqq \wedge(D_1, D_2) & C_2 \coloneqq id(D_3, D_3) \\ D_1 \coloneqq id(E_1, E_1) & D_2 \coloneqq \wedge(E_1, E_2) & D_3 \coloneqq \wedge(E_1, E_3) & \end{cases} \tag{1.7}$$

$$E(C') = \begin{cases} & & C' \coloneqq \wedge(C_1', C_2') & \\ & & C_1' \coloneqq \wedge(D_1', D_2') & C_2' \coloneqq id(D_3', D_3') \\ D_1' \coloneqq id(E_1, E_1) & D_2' \coloneqq \wedge(E_1, E_3) & D_3' \coloneqq \wedge(E_1, E_2) & \end{cases} \tag{1.8}$$

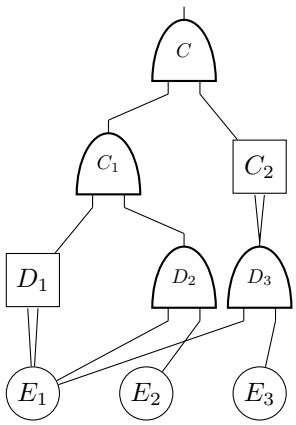


Figura 1.4: Representación física del circuito C.

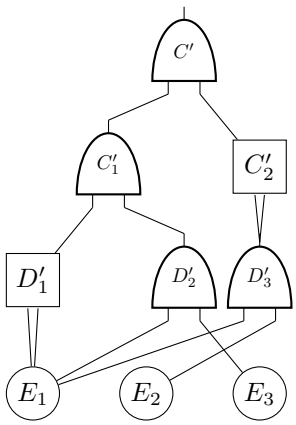


Figura 1.5: Representación física del circuito C'.

donde se han omitido de $E(C)$ y $E(C')$ las definiciones de E_1, E_2 y E_3 y todos sus subcircuitos. En estas condiciones es claro que $D_1 = D_1', D_2 = D_3'$ y $D_3 = D_2'$. Y por tanto es fácil ver que el orden inducido de manera recursiva vía un conjunto de funciones índices que cumplen que $D_1 < D_2 < D_3$ (y en consecuencia $D_1' < D_2' < D_3'$) sobre C y C' llevaría a que $E_1 < E_3 < E_2$ en el primer caso y que $E_1 < E_2 < E_3$ en el segundo.

Sin embargo, sería recomendable buscar unas funciones índices únivocamente determinadas para todos los niveles, de manera que dichas funciones interaccionen entre sí.

Teorema 1.4.5. *Para todo $n \in \mathbb{Z}$ existe una función índice determinada unívocamente por el orden entre circuitos y el orden entre puertas.*

Demostración. En primer lugar, analicemos los circuitos de profundidad 0. Así, sea $C \coloneqq in_i$ donde $0 \leq i < m$ de dicha profundidad y definamos $f_0(C) = i$. Por tanto, f_0 es una función índice basada en el orden entre circuitos.

Supongamos que existe ahora f_n en dichas condiciones, tratemos de definir f_{n+1}. Así, dado $C \coloneqq P(C_1, C_2)$, definamos $f_{n+1}(C) = \delta_m^2(n) - (\delta_m(n) - f_n(C_1))^2 + 2(f_n(C_2) - f_n(C_1)) - g(P)$, donde $g(P) = 1$ si $P = \vee$ y 0 en otro caso. La idea intuitiva de esta función se encuentra ilustrada en la figura 1.6. Veamos pues que f_{n+1} es biyectiva viendo que es sobreyectiva, ya que $|\mathcal{C}_{n+1}| = \delta_m(n+1) < \infty$.

En primer lugar, notemos que $f_{n+1}(\vee(C_1, C_2)) + 1 = f_{n+1}(\wedge(C_1, C_2))$ y que si $f_n(C_1) = 0$, entonces $P = id$ y $f_{n+1}(id(C_1, C_1) = 0$. Así, nuestro objetivo será construir una cadena de circuitos de profundidad $n+1$ tales que la diferencia de su evaluación vía f_{n+1} diste 1 y de longitud $\delta_m(n+1)$. Es claro que probado esto, la suprayectividad queda probada, y por ende, la biyectividad. Supongamos que tenemos el circuito $C \coloneqq P(C_1, C_2)$ y $f_{n+1}(C) = k$. Si $P = \vee$, entonces $\wedge(C_1, C_2)$ será el siguiente circuito de esta sucesión. Por tanto, solo queda analizar los circuitos de la forma $C \coloneqq P(C_1, C_2)$, con $P \in \{id, \wedge\}$.

Si $f_n(C_2) < \delta_m(n) - 1$, entonces sea C_2' el circuito de profundidad n que cumple que $f_n(C_2') = f_n(C_2) + 1$. Así, es fácil comprobar que $f_{n+1}(P(C_1, C_2)) + 1 = f_{n+1}(\vee(C_1, C_2'))$. En caso contrario, $f_n(C_2) = \delta_m(n) + 1$, ya que $|C_n| = \delta_m(n)$ y f_n está bien definida. Así, sea C_1' el circuito de profundidad n que cumple que $f_n(C_1') = f_n(C_1) + 1$. Entonces $f_{n+1}(id(C_1', C_1')) = f_{n+1}(C_1, C_2) + 1$.

Finalmente, contemos el número de circuitos que intervienen. Por el procedimiento empleado, es claro que es resultante de combinar el circuito de índice $f_n(C)$ con todos los circuitos con índice mayor o igual que este. Por tanto, el cardinal de este conjunto es: $\sum_{i=0}^{\delta_m(n)-1} (2 \cdot (\delta_m(n) - 1 - i) + 1) = 2\delta_m^2(n) - 2 \sum_{i=0}^{\delta_m(n)-1} i - \delta_m(n) = 2\delta_m^2(n) - \delta_m(n)(\delta_m(n) - 1) - \delta_m(n) = \delta_m^2(n) = \delta_m(n+1)$.

En definitiva, gracias a que esta cadena tiene longitud $\delta_m(n+1)$, el índice de su circuito minimal es 0 y la distancia entre dos elementos consecutivos es 1, es claro que f_{n+1} es sobreyectiva. □

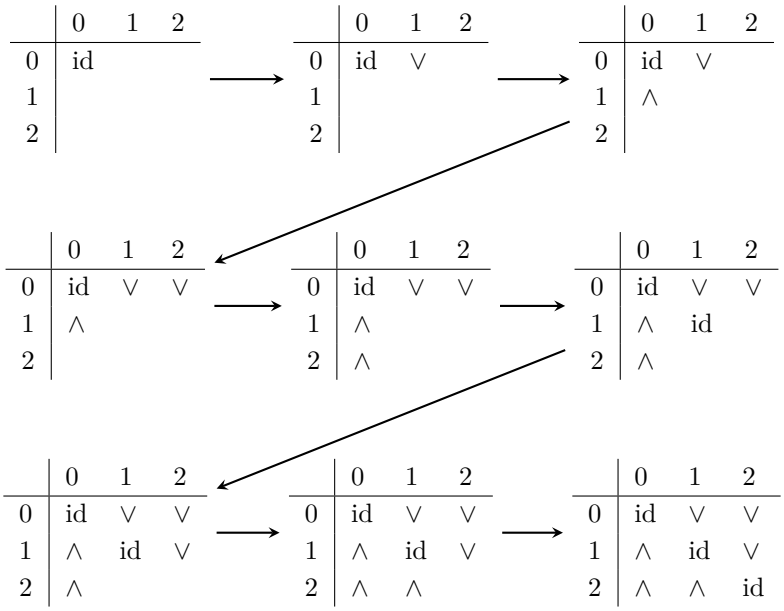


Figura 1.6: Visualización de la definición de índice f_1 cuando $m = 3$. Los números representan los valores de $f_0(C), C \in \mathcal{C}|_0$ y las flechas indican cuál es el siguiente circuito generado.

Corolario 1.4.6. *Dado $k \in \mathbb{Z}$ y una función índice $\hat{f}_n$ sobre un subconjunto de $S \subset \mathcal{C}_n$ de tamaño k existe una función índice $\hat{f}_{n+1}$ sobre el conjunto $T \subset \mathcal{C}_{n+1}$ generable por S compatible con f_n.*

Demostración. Por el lema 1.3.2, en T hay k^2. Así, sustituyendo $\mathcal{C}_n, \mathcal{C}_{n+1}$ y $\delta_m(n)$ por S, T y k respectivamente en la demostración del teorema 1.4.5 obtenemos la función $\hat{f}_{n+1}(C) = k^2 - (k - \hat{f}_n(C_1))^2 + 2(\hat{f}_n(C_2) - \hat{f}_n(C_1)) - g(P)$, la cual es índice en T. Finalmente, es inmediato ver que si $C < C'$ vía $\hat{f}_{n+1}$ entonces $C < C'$ vía f_{n+1}. □

Así, denotaremos a partir de ahora por $\mathcal{F}_n = \{f_0, ..., f_n\}$ al conjunto formado por las primeras n funciones índice calculadas en el teorema 1.4.5.

Corolario 1.4.7. *Sean $f_n, f_{n+1} \in \mathcal{F}_n$. El orden inducido por f_n en $\mathcal{C}_{n+1}$ coincide con el orden natural de f_{n+1}.*

Demostración. Considérese la cadena obtenida en el teorema 1.4.5 para circuitos de profundidad $n + 1$. Es inmediato comprobar que para cualesquiera dos circuitos consecutivos de dicha cadena, ambos cumplen que son menores según el orden inducido por f_n descrito en el lema 1.4.3. Así, el orden inducido por f_n coincide con el definido por f_{n+1}. □

Esta propiedad nos indica que estas funciones índices interaccionan adecuadamente. Así, $\mathcal{F} = \bigcup_{n\in\mathbb{N}} \mathcal{F}_n$ es el conjunto natural de funciones índice que ordena cada nivel de $\mathcal{C}$ en base al orden de los inputs y las puertas. Además, gracias a haber pasado por el teorema 1.4.5 podemos saber el orden de cada circuito dentro de su nivel con una fórmula más compacta.

Definición 1.4.8. Sea $C \coloneqq P(C_1, C_2)$ un circuito de profundidad $n > 0$ y sea $f_{n-1} \in \mathcal{F}$. Diremos que C está *bien conectado* si $f_{n-1}(C_1) \leq f_{n-1}(C_2)$. En caso contrario, diremos que C está mal conectado. Si C es un circuito de profundidad 0 siempre estará bien conectado.

Lema 1.4.9. *Todo circuito admite una representación bien conectada.*

Demostración. Ya que todo circuito de profundidad 0 está bien conectado, solo hay que estudiar los circuitos de profundidad n. Así, sea $C = P(C_1, C_2)$ circuito de profundidad n. Si $f_{n-1}(C_1) \leq f_{n-1}(C_2)$, hemos acabado, así que supongamos que no. Entonces, el circuito $C' = P(C_2, C_1)$ está bien conectado. □

Es decir, este lema nos dice que podemos construir siempre que queramos los circuitos con un mínimo orden. Sin embargo, nos gustaría tener los circuitos ordenados de manera que sus subcircuitos también lo estén y de manera global, también. Estos vagos conceptos los desarrollaremos con las siguientes definiciones.

Definición 1.4.10. Diremos que un circuito C está *ordenado lexicográficamente* si él y todos sus subcircuitos están bien conectados.

Aunque a priori el apellido lexicográfico parece que no está en absoluto relacionado con la definición, más adelante veremos el porqué del mismo.

Teorema 1.4.11. *Todo circuito de profundidad n admite una escritura lexicográfica, es decir, un reordenamiento en los argumentos de cada una de las puertas de manera que esté ordenado lexicográficamente.*

Demostración. Procederemos de manera aérea para mostrar claramente el resultado. Sea C un circuito de profundidad n y sea $C' \sqsubset C, C' \coloneqq P(C_1, C_2)$ de manera que C' no está bien conectado. Así escribamos $C' \coloneqq P(C_2, C_1)$. De esta manera, C' está bien conectado y el número de circuitos mal conectados se ha reducido en 1. En un número finito de pasos hemos acabado. □

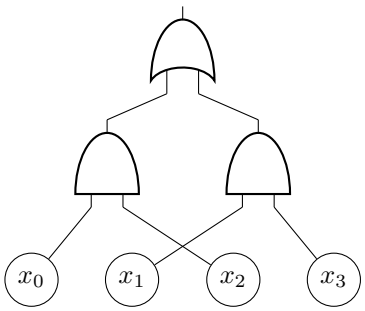
Sin embargo, esta demostración abusa de que tenemos explícitamente las funciones índices, que no nos importa calcular el valor de dicha función independientemente de su coste y que cambiar de orden los circuitos en la escritura tampoco tiene coste. En primer lugar, notemos que calcular la función índice es muy costoso según se incrementa la profundidad del circuito. Así, pongámonos en la situación computacional de intentar modelizar un circuito en estas condiciones con profundidades elevadas. Además, supongamos que guardamos los circuitos en listas de manera que en cada posición de la lista tenemos una puerta y dos índices indicando los circuitos que habría que unir. En estas circunstancias, el circuito $C \coloneqq P(C_1, C_2)$ sería la tupla $(repr(P), idx(C_1), idx(C_2))$, donde $idx(C)$

es el índice de la lista asociado a dicho circuito y $repr(P)$ un elemento con el que representar de manera unívoca la puerta y el circuito $C \coloneqq in_i$ se podría ver como la tupla (i).

En esta situación, la cual es ideal a nivel de uso de memoria (no se puede representar el circuito con menor cantidad de información), nos gustaría poder recorrer $\mathcal{C}_n$ sin saltarnos ningún circuito, recorriéndolos todos y en el menor tiempo posible. Para ello, la escritura de los circuitos de manera lexicográfica y el orden lexicográfico nos serán de gran utilidad.

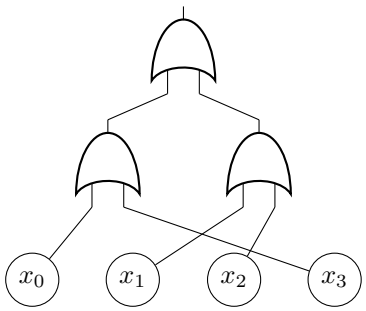
Antes de acabar el capítulo, mostremos el porqué del apellido *lexicográfico*. Visualizaremos los circuitos de manera *matricial*, esto es, cada profundidad indexa una fila y cada columna indexa un circuito de dicha profundidad.

Ejemplo 1.4.12. Tomemos los siguientes circuitos, los cuales mostramos en su representación física y matricial. Al circuito de la izquierda lo denominaremos C y al de la derecha C'.



$$\begin{bmatrix} (\vee,0,1) & & & \\ (\wedge,0,2) & (\wedge,1,3) & & \\ (0) & (1) & (2) & (3) \end{bmatrix}$$

Figura 1.7: Representación del circuito C.



$$\begin{bmatrix} (\vee,0,1) & & & \\ (\vee,0,3) & (\vee,1,2) & & \\ (0) & (1) & (2) & (3) \end{bmatrix}$$

Figura 1.8: Representación del circuito C'.

Así, según las definiciones anteriores, es claro que $C < C'$. Para visualizar correctamente la parte lexicográfica de dicho nombre simplificaremos un poco la notación matricial aún perdiendo expresividad e información de la misma de la siguiente manera:

$$\left[\begin{array}{cc|cc|cc} 0 & 1 & & & & \\ 0 & 2 & 1 & 3 & & \\ 0 & & 1 & & 2 & 3 \end{array}\right] \quad \text{y} \quad \left[\begin{array}{cc|cc|cc} 0 & 1 & & & & \\ 0 & 3 & 1 & 2 & & \\ 0 & & 1 & & 2 & 3 \end{array}\right]$$

. Observemos que a la notación matricial se le han añadido unas lineas verticales para separar visualmente en cada nivel el circuito asociado, con un fin nada más que aclaratorio. Con esta visualización parece razonable que $C < C'$ ya que $\left[\begin{array}{cc|} 0 & 1 & \end{array}\right] = \left[\begin{array}{cc|} 0 & 1 & \end{array}\right]$ pero $\left[\begin{array}{cc|cc} 0 & 2 & 1 & 3 \end{array}\right] < \left[\begin{array}{cc|cc} 0 & 3 & 1 & 2 \end{array}\right]$ utilizando los criterios lexicográficos habituales de vectores.

Esta observación es, de hecho, algo que sucede en general, incluyendo una tercera componente bivaluada para las puertas lógicas (la función de desempate g utilizada en el teorema 1.4.5). En consecuencia, parece razonable dotar con el sobrenombre de lexicográfico a este orden que hemos definido.

Tras enunciar y analizar todos los conceptos anteriores, indicamos que durante el resto del documento trabajaremos solo con circuitos en representación minimal ordenados lexicográficamente.

Capítulo 2

El algoritmo del índice

En este capítulo retomaremos los objetivos con los que cerrábamos el capítulo 1 explicando un algoritmo que recorra el espacio de circuitos en representación minimal ordenados lexicográficamente y bien conectados, el cual, abusando de nuevo de la notación, denotaremos simplemente por $\mathcal{C}$.

2.1. Resultados preliminares

Como $\mathcal{C}$ es un espacio con demasiados elementos, trataremos de buscar un algoritmo que dado n, recorra $\mathcal{C}_n$. Sin embargo, ya que $\mathcal{C}_n$ sigue siendo inabarcable, vamos a reducir levemente nuestras pretensiones.

Definición 2.1.1. Dado un circuito C de profundidad $n > 0$ denominamos *anchura* de C al valor $w(C) = \max\{|E_d(C)|, 1 \leq d \leq n\}$. Los circuitos de profundidad 0 tendrán anchura 0.

Lema 2.1.2. *La anchura de un circuito de profundidad n está entre 1 y 2^{n-1}.*

Demostración. La cota inferior se da gracias al ejemplo expuesto en el corolario 1.1.15. Por otra parte, la cota superior debe darse en un circuito en los que coincidan representación minimal y maximal. En este caso el circuito tiene $2^n - 1$ puertas y el mayor número de ellas en el nivel 1, en el que hay 2^{n-1}. □

Definición 2.1.3. Dado $n \in \mathbb{N}$ y dado $w \in \mathbb{N}, 1 \leq w \leq 2^{n-1}$ denotaremos por $\mathcal{C}_{n,w}$ a los circuitos de profundidad n y anchura w.

Así, una partición natural de $\mathcal{C}_n$ sería $\{\mathcal{C}_{n,1}, ..., \mathcal{C}_{n,w}\}$. Por tanto, podemos centrar todos nuestros esfuerzos en recorrer $\mathcal{C}_{n,w}$ ya que es claro que hecho esto, podemos recorrer cómodamente $\mathcal{C}_n$.

Destaquemos brevemente varias propiedades muy interesantes. En primer lugar, si en el nivel d hay k circuitos, por el lema 1.3.2 se pueden generar k^2 circuitos distintos. Además, podemos

emplear el corolario 1.4.6 para asegurarnos de que existe una función $\hat{f}_{d+1}$ que dota de orden a este conjunto de circuitos de manera que sea compatible con el índice, es decir, que genera el mismo orden que la función índice usual sobre los $\delta_m(d)$ circuitos distintos posibles.

Proposición 2.1.4. *Si en el nivel $d > 2$ de un circuito C hay k puertas, en el nivel $d+1$ hay entre $\lceil \frac{k}{2} \rceil$ y k^2 puertas y en el nivel $d-1$ entre $\lceil \sqrt{k} \rceil$ y $2k$.*

Demostración. Con respecto al nivel $d+1$, por el lema 1.3.2, es claro que hay a lo sumo k^2 puertas. Por otra parte, ya que toda puerta debe ser usada al menos una vez y hay k, la cota inferior se da inmediatamente.

Con respecto al nivel $d-1$, el número de puertas será mínimo si visto desde $d-1$, el número de puertas es máximo. Por tanto, habiendo k puertas en dicho nivel, en $d-1$ hay al menos $\lceil \sqrt{k} \rceil$. De manera análoga por la dualidad mínimo-máximo, se da la última de las cotas. □

Corolario 2.1.5. *Sea $d > 0$ y sea $C \in \mathcal{C}_{n,w}$, tal que $|E_d(C)| = k$. Entonces para todo nivel $j, 1 \leq j \leq d$ se da que $|E_j(C)| \geq \sigma^{d-j}(k)$, donde $\sigma(w) = \lceil \sqrt{k} \rceil$.*

Demostración. El resultado se da inmediatamente aplicando de manera reiterada la proposición 2.1.4. □

Corolario 2.1.6. *Sea $d > 0$ y sea $C \in \mathcal{C}_{n,w}$, si para todo $j > d$ se da que $|E_j(C)| < w$, entonces $|E_d(C)| \geq \eta^{d-1}(w)$, donde $\eta(w) = \lceil \frac{w}{2} \rceil$.*

Demostración. Por la definición de $\mathcal{C}_{n,w}$, en algún nivel se debe alcanzar la cota. Como no se alcanza en los niveles superiores a d, entonces en algún nivel entre 1 y d se debe alcanzar. Así, utilizando la proposición 2.1.4, se obtiene claramente el resultado. □

Lema 2.1.7 (Mínimo nivel)**.** *Sea C un circuito en $\mathcal{C}_{n,w}$. Entonces el mínimo nivel d tal que $|E_d(C)| = w$ es $d = \lceil \log_2(\text{máx}\{2, \log_m(w)\}) \rceil$.*

Demostración. De acuerdo al corolario 1.3.3 es claro que en el nivel d hay $\delta_m(d)$ posibles circuitos. Por tanto, si $|E_d(C)| = w$ es claro que $w \leq m^{2^d}$. Tomando logaritmos en ambos lados obtenemos que $2^{d-1} < \log_m(w) \leq 2^d$. Por otra parte $d \leq 1$, con lo que $2^d \leq 2$ y en consecuencia $2^d \leq \text{máx}\{2, \log_m(w)\}$. Volviendo a tomar logaritmos obtenemos que $d \leq \log_2(\{2, \log_m(w)\})$. Así, es claro que el nivel mínimo en el que se alcanza este valor es el descrito en el lema. □

Teorema 2.1.8. *El conjunto $\mathcal{C}_{n,w}$ no es vacío si y solo si dado $d = \lceil \log_2(\text{máx}\{2, \log_m(w)\}) \rceil$, se da que $\eta^{n-d}(w) = 1$.*

Demostración. Si $\eta^{n-d}(w) > 1$, entonces para cualquier circuito C de profundidad n se da que $|E_n(C)| \neq 1$ y eso es imposible ya que un circuito de profundidad n cumple que $|E_n(C)| = 1$. La otra implicación nos costará más esfuerzo y la desarrollaremos al final del capítulo; pero avanzamos su veracidad ya para garantizar al lector no estar trabajando sobre castillos en el aire. □

Así, aligeraremos la notación con las siguientes definiciones:

Definición 2.1.9. Dado un circuito $C \coloneqq P(C_1, C_2)$ de profundidad $n > 0$ llamaremos *cable izquierdo* o simplemente *izquierdo* a $l(C) = f_{n-1}(C_1)$ y llamaremos *cable derecho* o *derecho* a $r(C) = f_{n-1}(C_2)$. Además, los escribiremos como l y r respectivamente allá donde no haya lugar a confusión.

Definición 2.1.10. Dado un circuito $C \coloneqq P(C_1, C_2)$ de profundidad $n > 0$ llamaremos *tipo de puerta* o simplemente *tipo* al valor $t(P) = 1 - g(P)$, donde g es la función de desempate definida en el teorema 1.4.5; y este valor lo escribiremos como t si no induce a confusión.

Esta definición tiene sentido visualizando nuestras representación de los circuitos. El cable izquierdo de un circuito nos diría intuitivamente con cuál de los circuitos de profundidad $n-1$ debemos unirlo para formarlo, y análogamente el mismo significado sobre el cable derecho. Además, esta notación refleja el mismo significado si en vez de coger f_{n-1} tomamos $\hat{f}_{n-1}$ asociada a un subconjunto de $\mathcal{C}_{n-1}$. Finalmente, notemos que la fórmula descrita en el corolario 1.4.6 se puede expresar como:

$$\hat{f}_n(C) = l(2k - l) + 2(r - l) - (1 - t) \tag{2.1}$$

de lo que deducimos que un circuito se puede representar solamente utilizando los valores l, r y t.

Proposición 2.1.11. *Si partimos de un subconjunto S de $\mathcal{C}_{n-1}$ de tamaño k y $C' \in S$, los circuitos de la forma $C \coloneqq id(C', C')$ son los únicos que tienen por índice $i(2k - i), 0 \leq i < k$.*

Demostración. En este caso, en la fórmula 2.1 $r = l$ con lo que $\hat{f}_n(C) = k^2 - (k - l)^2 = l(2k - l)$ y $0 \leq l < k$ por la definición de índice. La unicidad se deriva de la biyectividad de la función índice. □

Corolario 2.1.12. *Todo circuito $C \coloneqq P(C_1, C_2)$ de profundidad $n > 1$ con índice $\hat{f}_n(C) = \alpha$ cumple que $l = k - \lfloor\sqrt{k^2 - \alpha}\rfloor, r = \frac{1}{2}(\alpha - l(2k - l)) + l$ y $g(P) \equiv \alpha - l(2k - l)$* mód 2.

Demostración. Por las propiedades del índice, existe un único l tal que $l(2k-l) \geq \alpha > (l+1)(2k-(l+1))$, y este es $k - \lfloor\sqrt{k^2 - \alpha}\rfloor$. Finalmente, los valores r y $g(P)$ se deducen trivialmente de la ecuación 2.1 al ser todos los valores enteros. □

Por tanto, si tenemos k circuitos de profundidad n y k' circuitos de profundidad $n+1$, a cada circuito de profundidad n le podemos asignar un único entero tal que $\hat{f}_n(C) = j$ y en consecuencia denotaremos por $C' \coloneqq t_i(l_i, r_i)$ al circuito $C' = P(C_1, C_2)$ tal que $l(C) = \hat{f}_n(C_1) = l_i$, $r(C) = \hat{f}_n(C_2) = r_i$ y $t(P) = t_i$ gracias al corolario 2.1.12. Cuando tengamos circuitos de más de una profundidad, indicaremos esta como segundo subíndice, es decir, $C_{j,n} \coloneqq t_{j,n}(l_{j,n}, r_{j,n})$. Este flagrante abuso de notación se verá prontamente recompensado en los resultados que siguen.

Por otra parte, ya que r, l y t definen el circuito C, podemos realizar otra representación matricial del mismo. Esta representación la denominaremos *representación matricial dispersa* para diferenciarla de la utilizada en el ejemplo 1.4.12, que renombraremos como *representación matricial compacta*. Finalmente, denotemos de ahora en adelante $\hat{f}_i, 0 \leq i \leq n$ a las funciones índices de $\mathcal{C}_{n,w}$ que se inducen de manera natural por $\mathcal{F}_n$.

Ejemplo 2.1.13. Mostremos las diferentes representaciones de un circuito tomando $m = 6, n = 4$ y $w = 3$.

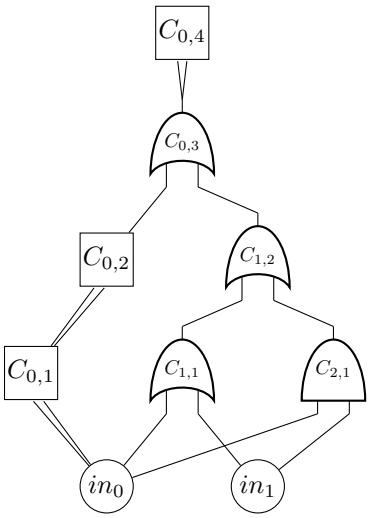


Figura 2.1: Representación física del circuito C.

$$\begin{bmatrix} (id,0,0) & & & \\ (\vee,0,1) & & & \\ (id,0,0) & (\vee,1,2) & & \\ (id,0,0) & (\vee,0,1) & (\wedge,0,1) \\ (0) & (1) & & \end{bmatrix}$$

(a) Representación matricial compacta.

$$\left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 1 & & & & & & \\ 0 & 1 & 0 & & & & & & \\ 0 & 0 & 1 & 1 & 2 & 0 & & & \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ & 0 & & & 1 & & & & \end{array}\right]$$

(b) Representación matricial dispersa.

Figura 2.2: Representación matricial del circuito C.

2.2. Índice y espacio cociente

El objetivo en este apartado será analizar cómo, dado un circuito C, obtener un circuito C' tal que se parezca mucho a C pero cuyo índice difiera en 1 frente a C. Sin embargo, aunque podríamos definir la cadena de circuitos que cumple esta propiedad, no es lo que vamos a hacer por una simple razón: cambiar el cableado de un circuito es más costoso que cambiar el tipo de una puerta. Por tanto, vamos a diseñar un algoritmo que reduzca el número de cambios de cable que hay que hacer cumpliendo algunas propiedades esenciales.

Definición 2.2.1. Denotaremos por $E|_d(C)$ al conjunto $\bigcup_{l=d}^{n} E_l(C)$.

Definición 2.2.2. Diremos que los circuitos C y C' están en la misma clase de equivalencia $C|_d$ si existe un renombramiento entre $E|_d(C)$ y $E|_d(C')$ y denotaremos por $\mathcal{C}_{n,w}|_d$ al conjunto formado por todas estas clases.

En otras palabras, $E|_d(C)$ consiste en restringir las ecuaciones que definen a C quedándonos

con los d últimos niveles y $C|_d$ consistiría en los circuitos que tienen los mismos r niveles iguales. Veámoslo con un ejemplo:

Ejemplo 2.2.3. Consideremos los circuitos con las siguientes representaciones matriciales:

$$\left[\begin{array}{ccc|ccc|ccc} 0 & 1 & 0 & & & & & & \\ 0 & 1 & 0 & 2 & 2 & 1 & & & \\ 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 1 \\ & 0 & & & 1 & & & 2 & \end{array}\right]$$

Figura 2.3: Representación matricial dispersa de C.

$$\left[\begin{array}{ccc|ccc|ccc} 0 & 1 & 0 & & & & & & \\ 0 & 1 & 0 & 2 & 2 & 1 & & & \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 & 2 & 0 \\ & 0 & & & 1 & & & 2 & \end{array}\right]$$

Figura 2.4: Representación matricial dispersa de C'.

Así, es claro que $C|_3 = C'|_3, C|_2 = C'|_2$ pero $C|_1 \neq C'|_1$ (y en consecuencia $C|_0 \neq C'|_0$); solo basta mirar fila a fila cada matriz para detectar esto. □

Observemos que una clase $C|_d$ puede estar contenida en $\mathcal{C}_{n,w}|_d$ pero también en $\mathcal{C}_{n,w'}|_d$. Esto quiere decir que hay un circuito C' en $\mathcal{C}_{n,w'}$ de manera que $C'|_d = C|_d$, o lo que es lo mismo, que $E|_d(C) = E|_d(C')$. Así, hay que tener especial cuidado si se cambia de clase para garantizar que esta nueva clase tiene algún circuito contenido en $\mathcal{C}_{n,w}$ (y generable por m inputs).

Notemos ahora algunas relaciones interesantes. En primer lugar, las clases $C|_d$ se pueden ver como elementos de $C|_{d+1}$ y la diferencia entre dos elementos de $C|_d$ es solamente los vectores $v = (l_{0,d}, r_{0,d}, ..., l_{p-1,d}, r_{p-1,d})$ y $g = (t_{0,d}, ..., t_{p-1,d})$, donde $p = |E_d(C)|$. En otras palabras, la diferencia entre dos clases son las puertas y cables del nivel d, y esta información queda plenamente recogida en los vectores v, g. De esta manera, podemos denotar de manera cómoda un circuito con la tupla de valores (t_j, l_j, r_j), la cual escribiremos como $t_j(l_j, r_j)$ para replicar la notación que habíamos seguido hasta ahora.

A su vez, las clases de $\mathcal{C}_{n,w}|_d$ se pueden agrupar entre aquellas que comparten el vector v. A estas clases las denotaremos $\hat{C}|_d$, al conjunto de las mismas la denotaremos $\hat{\mathcal{C}}_{n,w}|_d$ y diremos que v es el vector asociado a la misma. En consecuencia, tenemos el siguiente diagrama conmutativo, donde las funciones π consisten en la proyección sobre el conjunto imagen.

$$\begin{array}{ccc} \mathcal{C}_{n,w}|_{d+1} & \xrightarrow{\pi} & \hat{\mathcal{C}}_{n,w}|_{d+1} \\ \pi\uparrow & & \pi\uparrow \\ \mathcal{C}_{n,w}|_d & \xrightarrow{\pi} & \hat{\mathcal{C}}_{n,w}|_d \end{array}$$

Figura 2.5: Diagrama conmutativo entre los distintos conjuntos de clases.

Finalmente, hagamos notar que si $\hat{C}|_d$ es una clase de $\hat{\mathcal{C}}_{n,w}|_d$, el índice es un orden sobre el conjunto de elementos $C|_d \in \mathcal{C}_{n,w}|_d$ que son miembros de $\hat{C}_d$ y, por otra parte induce un orden sobre los elementos de $\hat{\mathcal{C}}_{n,w}|_d$. Por ello, vamos a aprovechar la función índice para recorrer $\mathcal{C}_{n,w}|_d$ por partes y en cada parte seguir el orden del índice aunque globalmente no sea así.

2.3. Incremento del tipo

Supongamos que estamos dentro de una clase $\hat{C}|_d$, donde $p = |E_d(C)|$ y $v = (l_0, r_0, ..., l_{p-1}, r_{p-1})$ es el vector asociado a la clase.

Proposición 2.3.1. *Dado v vector asociado a $\hat{C}|_d$, el vector $g = (t_0, ..., t_p)$ de la clase $C|_d$ dentro de $\hat{C}|_d$ cumple las siguientes propiedades:*

- *Si $l_j = r_j$, entonces $t_j = 1$, $0 \leq j < p$.*
- *Si $l_j = l_{j+1}, r_j = r_{j+1}$, $0 \leq j < p-1$, entonces $t_j = 0$ y $t_{j+1} = 1$.*

Demostración. Si $l_j = r_j$, entonces la puerta asociada es id, con lo que $t_j = 1$. Por otra parte, si $l_j = l_{j+1}$ y $r_j = r_{j+1}$, entonces no puede darse que $l_j = r_j$ debido a estar en representación minimal, ni que $P_j = P_{j+1}$, con lo que debido al índice obtenemos que $P_j = \vee$ y $P_{j+1} = \wedge$ y esto demuestra inmediatamente la proposición. □

A los valores de t_j del vector de tipos g que no estén determinados por la proposición 2.3.1 los denominaremos *tipos libres* o *libres*, y los denotaremos en el siguiente algoritmo por τ_j. Además, tomaremos $\gamma = (\tau_0, ..., \tau_T)$ como el vector de tipos libres.

Corolario 2.3.2 (Tipo mínimo)**.** *Sea $C|_d$ mínimo dentro de $\hat{C}|_d$ y sea $g = (t_0, ..., t_p)$ el vector de puertas asociado a $C|_d$. Entonces γ es el vector nulo.*

Demostración. La demostración es inmediata ya que cualquier otro valor de γ haría que el índice de la clase $C'|_d$ con dicho valor de γ fuese mayor. □

Proposición 2.3.3. *Dado v asociado a $\hat{C}|_d$ y dado $g = (t_0, ..., t_{p-1})$ asociado a $C|_d$ tenemos la siguiente alternativa:*

- *O bien existe un único $g' = (t'_0, ..., t'_{p-1})$ asociado a otra clase $C'|_d$ con mismo vector v tal que el índice inducido de $C'|_d$ dista 1 del de $C|_d$,*
- *O bien el índice inducido en $C|_d$ dentro de $\hat{C}|_d$ es máximo.*

Demostración. Sin pérdida de generalidad podemos quedarnos con el vector $\gamma = (\tau_0, ...\tau_T)$ ya que, por la proposición 2.3.1, los valores que no sean libres están determinados. Entonces, si vemos γ como un número en binario, es claro que el vector asociado al número $\gamma + 1$ es el único vector que podría cumplir la primera alternativa. Además, para que $\gamma + 1$ no esté bien definido tendría que darse que $\gamma \equiv 2^T - 1$, y en este caso es claro que el índice de $C|_d$ es máximo. □

Ejemplo 2.3.4. Mostraremos de una manera visual en qué se traduce la proposición 2.3.3. Se muestran en negro los elementos l_j, r_j de una clase $C|_{n-3}$ (supuesto que $n > 3$), así como los tipos

t_j de $C|_{n-2}$. Además, se muestran en rojo los elementos t_j que no son libres, en ámbar los libres tal que su tipo es 1 y en verde los que tienen $t_j = 0$.

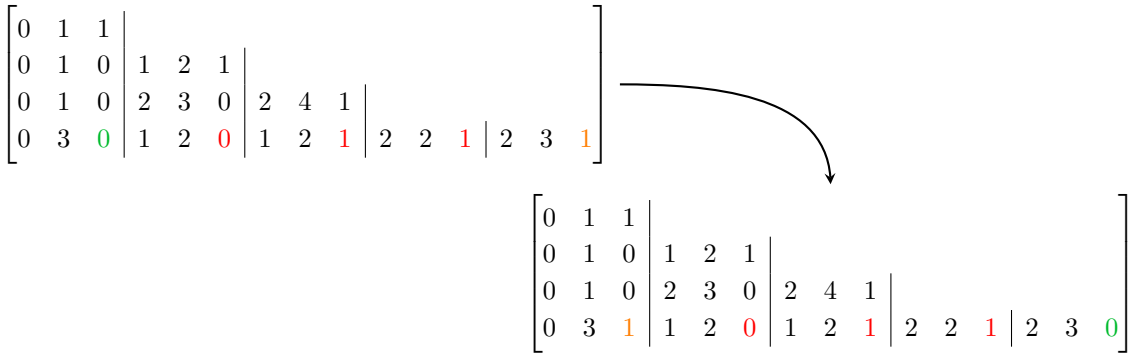


Figura 2.6: Obtención de $C'|_{n-3}$ a partir de $C|_{n-3}$ vía la proposición 2.3.3.

Se puede ver que la clase $C'|_{n-3}$ obtenida preserva los valores de los tipos no libres, y con los libres se comporta como un contador binario. Además, el color ámbar otorgado a los tipos indica intuitivamente que dicho valor no es incrementable. Por tanto, el índice de la clase será máximo cuando no haya ningún tipo de color verde; no podremos avanzar.

Algoritmo 2.3.5 (Incremento del tipo)**.** *Dado una clase $C|_d$ de $\hat{C}_k$ con vector asociado g, devuelve la clase $C'|_d$ cuyo vector asociado es g' y de índice* 1 *mayor que el de $C|_d$ o detecta que es imposible (y devuelve None).*

Algoritmo 1 Incremento del tipo

Require: $g = [t_0, ..., t_{p-1}]$
  $j \leftarrow p - 1$
  while $j \geq 0 \wedge (t_j$ no es libre $\vee\ t_j = 1)$ **do**
    if t_j es libre **then**
      $t'_j \leftarrow 0$ ▷ Si es libre entonces $t_j = 1$ y si existe g', $t'_j = 0$.
    else
      $t'_j \leftarrow t_j$ ▷ Si no es libre y existe g', $t'_j = t_j$.
    end if
    $j \leftarrow j - 1$
  end while
  if $j = -1$ **then**
    return $None$ ▷ No existe g'.
  else
    $t'_j \leftarrow 1$ ▷ Existe g' y t'_j es libre, así que $t'_j = 1$.
  end if
  return $g' = [t_0, ...t_{j-1}, t'_j, ..., t'_{p-1}]$

Demostración. La corrección de este algoritmo es inmediata gracias a la proposición 2.3.3. □

Como se puede ver, el algoritmo tiene un coste máximo de $\mathcal{O}(p)$, pero, por ser *de facto* un contador binario, el coste amortizado lo rebaja a $\mathcal{O}(T+1)$, donde T es el número de libres en $\hat{C}|_d$. En definitiva, hemos conseguido recorrer la clase $\hat{C}|_d$ incrementando en cada paso el vector tipo un poco, de ahí el nombre del algoritmo.

2.4. La propiedad del buen prefijo

Nuestro siguiente objetivo será, fijada la clase $\hat{C}|_{d+1}$ recorrer las clases de $\hat{\mathcal{C}}_{n,w}|_d$ siguiendo el orden del índice. Para ello nos valdremos de la *propiedad del buen prefijo*, la cual expondremos en esta sección. Así, supongamos que $d > 0$ y denotemos $p = |E_d(\hat{C})|$ y $k = |E_{d-1}(\hat{C})|$; y tomemos una clase $\hat{C}|_d$ y su vector v asociado, el cual tiene tamaño $2p$. Tengamos en cuenta que el objetivo de esta sección es buscar una clase $\hat{C}'|_d$ tal que $|E_d(\hat{C}')| = p$, $|E_{d-1}(\hat{C})| = k$ de manera que podamos pasar *cómodamente* de $\hat{C}|_d$ a $\hat{C}'|_d$.

Definición 2.4.1. Diremos que $j \in \mathbb{Z}$ es un *valor válido* si $0 \leq j < k$.

Definición 2.4.2. Dado un vector $v' = (l_0, r_0, ...l_j, r_j)$ denotaremos por $S_j, -1 \leq j < p$ al *conjunto de restantes* $S_j = \mathbb{Z}_k \setminus \{l_i, r_i \mid 0 \leq i \leq j\}$ y denominaremos a v' el *vector complemento* de S_j.

Intuitivamente, dada una colección de valores $\{l_0, r_0, ...l_{j-1}, r_{j-1}\}$, S_j representa los índices de los circuitos que faltan por añadir a un vector v' asociado a alguna clase $\hat{C}|_k$ de manera que esté

bien formada, esto es, que utilice todos los circuitos del nivel inferior al menos una vez. Por ello, $S_{-1} = \{i \mid 0 \leq i < k\}$ y $S_{p-1} = \emptyset$. Normalmente, y mientras no se diga nada, tomaremos como vector complemento el $v_j = (l_0, r_0, ..., l_j, r_j)$, resultante de restringirnos a las $2(j+1)$ primeras componentes del vector v asociado a $\hat{C}|_d$.

Lema 2.4.3. *Si v es un vector de tamaño $2p$ asociado a una clase $\hat{C}|_d$, entonces $S_p = \emptyset$.*

Demostración. Si $S_p \neq \emptyset$, entonces no puede estar asociado a ninguna clase de $\hat{C}|_d$ ya que en ese caso faltarían circuitos de $E_{d-1}(\hat{C})$ por utilizar. □

Definición 2.4.4. Diremos que $v' = (v'_0, ..., v'_r)$ es un *prefijo* de $v = (v_0, ..., v_s)$, con $r \leq s$ si $v'_j = v_j$ para todo $0 \leq j \leq r$ y llamaremos número de *huecos* a $h(v, v') = s - r$.

Además, dado $v = (v_0, ..., v_r)$ denotaremos por $v + \alpha$ al vector $(v_0, ..., v_r, \alpha)$.

Lema 2.4.5 (Falta de huecos)**.**

1. *Si $v' = (l_0, r_0, ...l_j, r_j)$ es un prefijo de v y si α es un valor válido tal que $v' + \alpha$ sea prefijo de un vector v'' de tamaño $2p$ asociado a alguna clase $\hat{C}'|_d$, entonces $|S_j \setminus \{\alpha\}| \leq h(v, v' + \alpha)$.*
2. *Si $v' = (l_0, r_0, ...l_j, r_j, l_{j+1})$ es un prefijo de v y si α es un valor válido tal que $v' + \alpha$ sea prefijo de un vector v'' de tamaño $2p$ asociado a una clase $\hat{C}'|_d$, entonces $|S_j \setminus \{l_{j+1}, \alpha\}| \leq h(v, v' + \alpha)$.*

Demostración.

1. Si se diese que $|S_j \setminus \{\alpha\}| > h(v, v' + \alpha)$, entonces de ninguna manera el vector v'' cumpliría que $S_p = \emptyset$, lo cual contradiría el lema 2.4.3.
2. Análoga al apartado anterior.

□

Lema 2.4.6 (Exceso de huecos)**.** *Sea $v' = (l_0, r_0, ...l_j, r_j)$ un prefijo de v y sean $\lambda \leq \rho$ dos valores válidos. Si $v' + \lambda + \rho$ es prefijo de un vector v'' de tamaño $2p$ asociado a alguna clase $\hat{C}'|_d$, entonces $2(k^2 - 1 - \hat{f}_n(\hat{C})) \geq h(v, v' + \lambda + \rho)$, donde $\hat{C} \coloneqq \tau(\lambda, \rho)$ y $\tau = 1$ si $\lambda = \rho$ y 0 en otro caso.*

Demostración. Por la proposición 2.1.4, solo hay $k^2 - 1 - \hat{f}_n(\hat{C})$ circuitos generables a partir de los k existentes que tengan índice mayor que $\hat{C}$. Además, cada circuito emplea 2 argumentos, con lo que el máximo número de valores que se pueden añadir al prefijo $v' + \lambda + \rho$ es $2(k^2 - 1 - \hat{f}_n(\hat{C}))$. Y este valor debe ser al menos $h(v, v' + \lambda + \rho)$. Nótese que se toma $\tau = 0$ si $\lambda \neq \rho$ para que el circuito $1(\lambda, \rho)$ pueda considerarse también. □

Tengamos en cuenta que el nombre de los lemas 2.4.5 y 2.4.6 vienen dados por la negación de los mismos. Así, si no hay suficientes o hay demasiados huecos, es claro que al añadir los valores válidos no podemos obtener un vector v'' valido para alguna clase $\hat{C}|_d$.

Lema 2.4.7 (Test izquierdo)**.** *Sea $v' = (l_0, r_0, ...l_j, r_j)$ un prefijo de v y sea λ un valor válido. Si $v' + \lambda$ es prefijo de un vector v'' asociado a alguna clase $\hat{C}|_d$ entonces o bien $\hat{S}_j = S_j \setminus \{\lambda\} = \emptyset$ o bien $\lambda < \text{mín}\{\hat{S}_j\}$.*

Demostración. Si $\hat{S}_j \neq \emptyset$, entonces sea $\rho = \text{mín}\, \hat{S}_j$. Como $\lambda \neq \rho$, entonces $\lambda < \rho$, ya que en caso contrario el circuito $C \coloneqq P(C_1, C_2)$ tal que $l = \lambda, r = \rho$ cumpliría que $l > r$ y entonces no estaría bien conectado; lo cual se contradice con las propiedades de nuestro conjunto $\mathcal{C}_n$. □

Gracias a estos lemas, hemos podido apreciar propiedades que deben cumplir los valores λ y ρ con los que queremos extender nuestro prefijo. Con los siguientes teoremas veremos que estas propiedades no solo son necesarias sino que también son suficientes.

Teorema 2.4.8 (Propiedad del buen prefijo - derecho)**.** *Sea v el vector asociado a $\hat{C}|_d$, sea $v' = (l_0, r_0, ..., l_{j+1})$ y sea $\rho \geq l_{j+1}$ un valor válido tal que $1(l_j, r_j) < \tau(l_{j+1}, \rho)$, con $\tau = 1$ si $l_{j+1} = \rho$ y 0 en otro caso. Entonces $v' + \rho$ es un prefijo de algún vector v'' asociado a alguna clase $\hat{C}'|_d$ si y solo si se dan las siguientes condiciones:*

- $|S_j \setminus \{l_{j+1}, \rho\}| \leq h(v, v' + \rho)$.
- $2(k^2 - 1 - \hat{f}_n(\tau(l_{j+1}, \rho))) \geq h(v, v' + \rho)$, *donde $\tau = 1$ si $l_{j+1} = \rho$ y 0 en otro caso.*

Demostración.
$\Longrightarrow$ Es consecuencia inmediata de los lemas 2.4.5 y 2.4.6.

$\Longleftarrow$ Si $2(k^2 - 1 - \hat{f}_n(\tau(l_{j+1}, \rho))) \geq h(v, v' + l_{j+1} + \rho)$ entonces hay al menos una colección de valores $V = \{l_i, r_i \mid j + 1 < i < p\}$ ordenada vía el índice de manera que todos los circuitos de V sean distintos. Por otra parte, al ser v' asociado a v, se cumple que o bien $S_j \setminus \{l_{j+1}, \rho\}$ es vacío o bien $l_{j+1} < \text{mín}\, S_j \setminus \{l_{j+1}\}$.

Si es vacío, tomemos $V' = V$ y procedamos al el párrafo siguiente. Si no, ya que $|S_j \setminus \{l_{j+1}, \rho\}| \leq h(v, v' + \rho)$, podemos tomar los circuitos $(s_i, s_{i+1}), s_i, s_{i+1} \in S_j$, con S_j ordenado vía el índice (tomando (s_{i+1}, s_{i+1}) en el caso de que sean impares) y sustituir estos $cs = \lfloor \frac{1}{2}(|S_j \setminus \{l_{j+1}, \rho\}| + 1) \rfloor$ circuitos por cualesquiera cs parejas de valores de V obteniendo V'.

De esta manera, ordenando V' vía el índice y tomando $v'' = v' + \rho + V'$, es claro que v'' está bien formado ya que a lo sumo hay un par de índices $i, i + 1$ tales que $(l_i, r_i) = (l_{i+1}, r_{i+1})$ para cada tupla (l, r) y utiliza todos los circuitos de $E_{d-1}(\hat{C})$. En consecuencia, es un vector asociado a alguna clase $\hat{C}'|_d$ y $v' + \rho$ es prefijo de él. □

Teorema 2.4.9 (Propiedad del buen prefijo - izquierdo)**.** *Sea v el vector asociado a $\hat{C}|_d$, sea $v' = (l_0, r_0, ..., l_j, r_j)$, con $j < p - 1$ y sean λ, ρ valores válidos tales que $\lambda \leq \rho$ y tales que $1(l_j, r_j) < \tau(\lambda, \rho)$, con $\tau = 1$ si $\lambda = \rho$ y 0 en otro caso. Entonces $v' + \lambda + \rho$ es un prefijo de algún vector v'' asociado a alguna clase $\hat{C}'|_d$ si y solo si se dan todas las condiciones siguientes:*

- $|S_j \setminus \{\lambda, \rho\}| \leq h(v, v' + \lambda + \rho)$.

- $2(k^2 - 1 - \hat{f}_n(\tau(\lambda, \rho))) \geq h(v, v' + \lambda + \rho)$, *donde* $\tau = 1$ *si* $\lambda = \rho$ *y* 0 *en otro caso.*
- *O bien* $S_j \setminus \{\lambda\} = \emptyset$, *o bien* $\lambda \leq \min S_j$.

Demostración.
$\Longrightarrow$ Es consecuencia inmediata de los lemas 2.4.5, 2.4.6 y 2.4.7.

$\Longleftarrow$ Si $2(k^2 - 1 - \hat{f}_n(\tau(\lambda, \rho))) \geq h(v, v' + \lambda + \rho)$ entonces hay al menos una colección de valores $V = \{l_i, r_i \mid j + 1 < i < p\}$ ordenada vía el índice de manera que todos los circuitos de V sean distintos. Por otra parte, por la tercera condición, o bien $S_j \setminus \{\lambda\} = \emptyset$, o bien $\lambda \leq \text{mín}\{S_j\}$.

Así, definido V y gracias a la tercera condición podemos proceder de manera análoga al teorema anterior para obtener un V' tal que $v'' = v' + \lambda + \rho + V'$ sea un vector asociado a alguna clase $\hat{C}'|_d$ y en consecuencia, $v' + \lambda + \rho$ es prefijo de v''. □

Corolario 2.4.10 (Mínimo ρ)**.** *En las condiciones del teorema 2.4.9, si* $h(v, v' + \lambda) > |S_j \setminus \{\lambda\}|$, *podemos tomar* $\rho = \lambda$. *Si no, y supuesto que* $\{s \in S_j \mid s > \lambda\} \neq \emptyset$, *el valor* $\rho = \text{mín}\{s \in S_j \mid s > \lambda\}$ *satisface el teorema para dicho* λ. *Además, este* ρ *es el mínimo valor obtenible que satisface el teorema para dicho valor de* λ.

Demostración. En primer lugar notemos que la tercera condición del teorema no se ve afectada por este resultado al no depender de ρ. Así, si $h(v, v'+\lambda) > |S_j \setminus \{\lambda\}|$, entonces $h(v, v'+\lambda+\lambda) \geq |S_j \setminus \{\lambda\}|$, lo cual equivale a la primera condición. Además, es claro que no hay menor ρ que permita que $\tau(\lambda, \rho)$ esté bien conectado. Finalmente, por la definición del índice, la cota superior de los huecos cumple que $2(k^2 - 1 - \hat{f}_n(\tau(\lambda, \rho))) \leq 2(k^2 - 1 - \hat{f}_n(\tau(\lambda, \lambda)))$. Es decir, si hay algún valor ρ que posibilita que se cumpla el teorema para dicho v' y dicho λ, este debe ser $\rho = \lambda$.

Sin embargo, si $h(v, v' + \lambda) \leq |S_j \setminus \{\lambda\}|$, entonces no se cumple la primera condición con dicho ρ. De hecho, no se cumplirá para ningún ρ que no esté en $S_j \setminus \{\lambda\}$. Por ello, ρ debe estar en $S_j \setminus \{\lambda\}$. Además, para que esté bien conectado, $\rho \geq \lambda$, por lo que, utilizando la segunda condición de nuevo, en el caso de haber un ρ que cumpla las condiciones del teorema para este λ, $\rho = \text{mín}\{s \in S_j \mid s > \lambda\}$ las cumplirá, con lo que esta definición de ρ lo hace mínimo. □

Notemos que si $h(v, v' + \lambda) \leq |S_j \setminus \{\lambda\}|$ y $\{s \in S_j \mid s > \lambda\} = \emptyset$, entonces hemos probado además que el vector $v' + \lambda$ no es prefijo de ningún v'' asociado a alguna clase $\hat{C}'|_d$.

Algoritmo 2.4.11 (Propiedad del buen prefijo)**.** *Dado* $v = (l_0, r_0, ..., l_{p-1}, r_{p-1})$ *vector de* $\hat{C}|_d$, j *entero tal que* $0 \leq j < p$, S_j *y* ρ *o* λ *según proceda, devuelve si* $v' = (l_0, r_0, ..., l_j, r_j, \lambda, \rho)$ *cumple la propiedad de buen prefijo.*

Algoritmo 2 Propiedad del buen prefijo

Require: S_j, λ o ρ y $h(v, v')$.
  if λ no es $None$ **then**
    Comprobar el lema Test izquierdo ▷ Si no se da, devuelve false.
    Obtener ρ usando el corolario Mínimo ρ ▷ Si no existe, toma el valor $None$.
  else
    $\lambda \leftarrow l_{j+1}$ ▷ Si $\rho < l_{j+1}$, toma el valor $None$.
  end if
  if λ o ρ es $None$ **then**
    return false
  else
    Comprobar los lemas Falta de huecos y Exceso de huecos ▷ Si no se dan, devuelve false.
    return true ▷ Se cumplen todas las condiciones del buen prefijo.
  end if

Demostración. La demostración se sigue de los teoremas 2.4.8, 2.4.9 y del corolario 2.4.10. □

Nótese que dado j, el coste de obtener $h(v, v')$ suele ser constante en la mayoría de los lenguajes de programación. Así, suponiendo que S es un conjunto implementado con un montículo que tenemos calculado previamente, el coste de validar los lemas 2.4.5 y 2.4.7 está en $\mathcal{O}(\log(p))$ al igual que el coste del corolario 2.4.10. Finalmente, el coste de comprobar el lema 2.4.6 es constante, con lo que el coste de este algoritmo es del orden de $\mathcal{O}(\log(p))$.

Recapitulando brevemente esta sección: dado un vector v asociado a una clase $\hat{C}|_d$ y un valor λ (respectivamente ρ) hemos obtenido un test para saber si existe un prefijo de tamaño $2j$ (respectivamente $2j+1$) de v de manera que también lo sea de un vector v' distinto.

2.5. Incremento del cableado

Gracias a la propiedad del buen prefijo y dado un vector v asociado a una clase $\hat{C}|_d$, trataremos de obtener otro vector v' asociado a otra clase $\hat{C}'|_d$ de manera que $\hat{C}|_{d+1} = \hat{C}'|_{d+1}$. Además, mostraremos cómo poder hacerlo de manera que el índice entre ambas clases diste 1. En esta sección pondremos el foco en cómo obtener $\hat{C}'|_d$ tal que $|E_d(\hat{C})| = |E_d(\hat{C}')| = p$ y $|E_{d-1}(\hat{C})| = |E_{d-1}(\hat{C}')| = k$, si esto es posible; donde p y k se definen como en la sección anterior.

Lema 2.5.1 (Siguiente derecho)**.** *Sea* $v_j^r = (l_0, r_0, ..., l_j)$. *Si ni* $\rho' = r_j + 1$ *ni* $\rho'' = \text{mín}\{s \in S_{j-1} \mid s > r_j\}$ *(supuesto que* $\{s \in S_{j-1} \mid s > r_j\} \neq \emptyset$*) son valores válidos con los que se cumple la propiedad del buen prefijo, entonces no existe ningún* $\rho > r_j$ *que lo permita.*

Demostración. En primer lugar notemos que si $r_j + 1$ no es válido, no existe $\rho > r_j$ que sea válido, con lo que sin pérdida de generalidad supongamos que $r_j + 1$ es válido. Además, por ser v_j^r prefijo de v, se da que $r_j \geq l_j$.

Así, razonemos por contradicción. Sea ρ distinto a ambos valores. Si ρ' no cumple las condiciones del teorema, es porque o bien $2(k^2 - 1 - \hat{f}_n(0(l_j, \rho'))) < h(v, v_j^r + \rho')$ o bien $|S_{j-1} \setminus \{l_j, \rho'\}| > h(v, v_j^r + \rho')$. Si es porque se da que $2(k^2 - 1 - \hat{f}_n(0(l_j, \rho'))) < h(v, v_j^r + \rho')$, es claro que no existe ningún ρ admisible, ya que $\hat{f}_n(0(l_j, \rho')) < \hat{f}_n(0(l_j, \rho))$ y en consecuencia el lema 2.4.6 no se cumplirá para ningún valor ρ.

Por tanto, podemos suponer que lo que acontece es que $|S_{j-1} \setminus \{l_j, \rho'\}| > h(v, v_j^r + \rho')$. Así, para cualquier ρ tal que $|S_{j-1} \setminus \{l_j, \rho\}| = |S_{j-1} \setminus \{l_j, \rho'\}|$ se va a dar que $|S_{j-1} \setminus \{l_j, \rho\}| > h(v, v_j^r + \rho)$; con lo que de existir dicho ρ, este tiene que estar en $S_{j-1} \setminus \{l_j\}$. Además, $\rho > r_j$, con lo que debe estar en $\hat{S}_{j-1} = \{s \in S_{j-1} \setminus \{l_j\} \mid s > r_j\}$. Si $\hat{S}_{j-1} = \emptyset$, hemos acabado la demostración. Si no, podemos tomar ρ'' como en el enunciado del teorema, y, ya que $\rho \geq \rho'' \geq \rho'$, podemos razonar como en el párrafo anterior para deducir que debe darse que $|S_{j-1} \setminus \{l_j, \rho''\}| > h(v, v_j^r + \rho'')$ si existe dicho valor ρ. Así, ya que $|S_{j-1} \setminus \{l_j, \rho''\}| = |\hat{S}_{j-1}| + 1 = |S_{j-1} \setminus \{l_j, \rho\}|$, es claro que ρ incumple el lema 2.4.5. □

Este resultado es muy importante ya que nos dice que dado v' de tamaño impar, solo nos basta con probar dos valores de ρ para ver si $v' + \rho$ es prefijo de un vector v'' asociado a alguna clase $\hat{C}'|_d$.

Lema 2.5.2 (Siguiente izquierdo)**.** *Sea $v_j^l = (l_0, r_0, ..., l_{j-1}, r_{j-1})$. Si ni $\lambda' = l_j + 1$ ni $\lambda'' = \text{mín}\{s \in S_{j-1} \mid s > l_j\}$ (supuesto que $\{s \in S_{j-1} \mid s > l_j\} \neq \emptyset$) son valores válidos con los que se cumple la propiedad del buen prefijo independientemente del ρ, entonces no existe ningún $\lambda > l_j$ que lo permita.*

Demostración. Por el corolario 2.4.10, es claro que solo tenemos que probarlo con dos valores candidatos a ρ. La demostración es análoga al lema 2.5.1 teniendo en cuenta además que hay que tener en cuenta la satisfacibilidad del lema 2.4.7. □

Lema 2.5.3 (Siguiente izquierdo simplificado)**.** *Sea $v_j^l = (l_0, r_0, ..., l_{j-1}, r_{j-1})$. Si $\lambda' = l_j + 1$ no es un valor válido con el que se cumple la propiedad del buen prefijo independientemente del ρ, entonces no existe ningún $\lambda > l_j$ que lo permita.*

Demostración. Por el lema 2.5.2, nos basta probar que si $\lambda' = l_j + 1$ no es un valor admisible para el vector v', entonces $\lambda'' = \text{mín}\{s \in S_{j-1} \mid s > l_j\}$ no lo es tampoco para v', ya que en caso de que se dé que $\{s \in S_{j-1} \mid s > l_j\} = \emptyset$ dicho lema nos garantiza el resultado.

Así, razonemos por contradicción: supongamos que existe un $\rho \geq \lambda''$ tal que $v' + \lambda'' + \rho$ es prefijo de v''. Entonces, como $\rho \geq \lambda'' > \lambda'$, es claro que se cumple el lema 2.4.6 para λ'', ρ y v'. Ya que se cumple el lema 2.4.7 para λ, puede suceder que $S_{j-1} = \emptyset$, o bien $S_{j-1} = \{\lambda''\}$, o bien $\lambda'' \leq \text{mín}\, S_{j-1}$. En el primer caso, es claro que se cumple dicho lema para λ', y en los otros dos podemos plantear la desigualdad $\lambda' < \lambda'' \leq \text{mín}\, S_{j-1}$; con lo que si λ'' cumple el lema 2.4.7, λ' también.

En consecuencia, tiene que suceder que $|S_{j-1} \setminus \{\lambda', \rho\}| > h(v, v' + \lambda' + \rho)$. Sin embargo, $\lambda' \notin S_{j-1}$, ya que si no no se cumpliría el test izquierdo para λ''. Por tanto, $|S_{j-1} \setminus \{\lambda', \rho\}| = |S_{j-1} \setminus \{\rho\}|$. Finalmente, se da la siguiente desigualdad: $h(v, v') + 2 = h(v, v' + \lambda' + \rho) < |S_{j-1} \setminus \{\lambda', \rho\}| = |S_{j-1} \setminus \{\rho\}| \leq |S_{j-1}| \leq h(v, v')$, o lo que es lo mismo $2 < 0$, que es una clara contradicción. □

Destaquemos que la existencia de una versión simplificada del lema 2.5.2 radica en el hecho de que $\lambda' \notin S_{j-1}$, cosa que no podemos garantizar para el valor ρ' del lema 2.5.1. Además, de manera colateral, obtenemos que $\lambda' \neq 0$ para ningún j. En consecuencia y gracias a estos dos lemas, podemos enunciar el siguiente algoritmo:

Algoritmo 2.5.4 (Prefijo mínimo)**.** *Dado $v = (l_0, r_0, ..., l_{p-1}, r_{p-1})$ vector de $\hat{C}|_d$, obtenemos el vector v' que sea prefijo de v'' y v'' diste 1 de v vía el índice si existe.*

Algoritmo 3 Prefijo mínimo

Require: $v = (l_0, r_0, ..., l_{p-1}, r_{p-1})$.
  $S \leftarrow \emptyset$
  $j \leftarrow p - 1$
  while $j > 0$ **do**
    $S \leftarrow S_{j-1} \setminus \{l_j\}$
    Obtener ρ'_j y ρ''_j con el lema Siguiente derecho.
    if $v_j^r + \rho'_j$ cumple la Propiedad del buen prefijo **then**
      return $v' = v_j^r + \rho'_j$
    else if $v_j^r + \rho''_j$ cumple la Propiedad del buen prefijo **then**
      return $v' = v_j^r + \rho''_j$
    end if
    $S \leftarrow S_{j-1}$
    Obtener λ'_j con el lema Siguiente izquierdo simplificado.
    if $v_j^r + \lambda'_j$ cumple la Propiedad del buen prefijo **then**
      return $v' = v_j^l + \lambda'_j + \hat{\rho}_j$ ▷ $\hat{\rho}_j$ es el valor de ρ obtenido con el lema Mínimo ρ para λ'_j.
    end if
    $j \leftarrow j - 1$
  end while
  $S \leftarrow S_{-1} \setminus \{l_0\}$
  Obtener ρ'_0 y ρ''_0 con el lema Siguiente derecho
  if $v_0^r + \rho'_0$ cumple la Propiedad del buen prefijo **then**
    return $v' = v_0^r + \rho'_0$
  else if $v_0^r + \rho''_j$ cumple la Propiedad del buen prefijo **then**
    return $v' = v_0^r + \rho''_0$
  else
    return None ▷ No existe el prefijo ya que mín $S_{-1} = 0$ y eso contradice el lema 2.5.3.
  end if

Demostración. En cada paso de la iteración, analizamos primero v_j^r y luego v_j^l. Por el lema 2.5.1, $\rho' \leq \rho''$ y son los mínimos valores que hacen que $v_j^r + \rho$ sea un prefijo de algún v'' en las condiciones requeridas. E igualmente sobre λ' y λ'' gracias al lema 2.5.3. Así, es claro que si para todo $j > 0$ el algoritmo no ha encontrado un prefijo, solo nos queda por analizar v_0^r, ya que v_0^l no puede generar un prefijo admisible: un vector v'' en estas condiciones debe empezar en 0. Además, por el orden de análisis de cada opción, es claro que de obtener un prefijo v', este es mínimo. □

Notemos que, si tenemos en una estructura de escritura y acceso constante el número de posiciones del vector que contienen al valor α, denominada número de *ocurrencias*, entonces podemos mantener una estructura de conjunto implementado con montículos de manera que la generación de S en cada etapa esté en $\mathcal{O}(\log(p))$, lo cual, sumado al coste del algoritmo 2.4.11, da lugar a que en el caso peor este algoritmo tiene coste en $\mathcal{O}(p \log(p))$.

Ejemplo 2.5.5. Mostraremos ahora de una manera visual cómo interpretar el algoritmo 2.5.4. Se muestra solamente la representación matricial dispersa de $\hat{\mathcal{C}}|_{n-2}$ (supuesto $n > 2$) por sencillez en la notación.

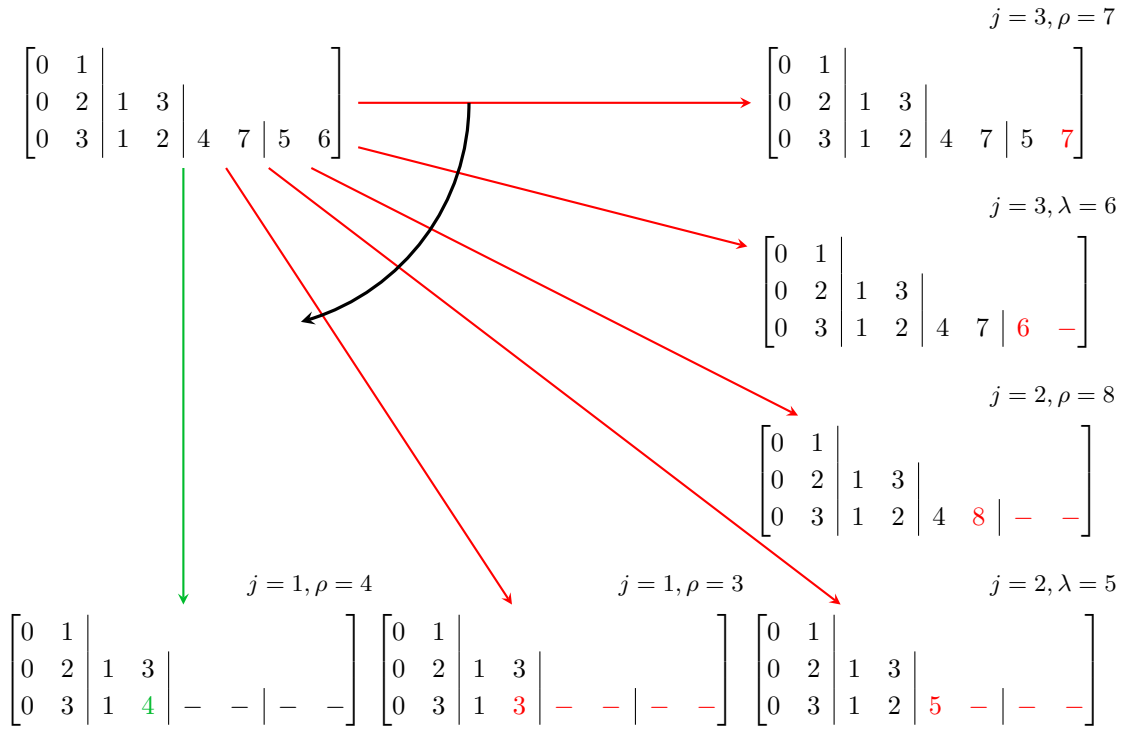


Figura 2.7: Ejecución del algoritmo 2.5.4 a partir de $\hat{C}|_{n-2}$.

Así, analicemos quién es el prefijo mínimo del vector v' de alguna clase $\hat{C}'|_{n-2}$ tal que $\hat{C}|_{n-2} < \hat{C}'|_{n-2}$ a partir de $v = \left[\begin{array}{cc|cc|cc|cc} 0 & 3 & 1 & 2 & 4 & 7 & 5 & 6 \end{array}\right]$.

En primer lugar, tomando el valor $j = 3$, se intenta probar $\rho = r_j + 1 = 7$. Sin embargo, el vector $v' + 7 = \left[0 \quad 3 \mid 1 \quad 2 \mid 4 \quad 7 \mid 5 \quad 7\right]$ no cumple el teorema 2.4.8 ya que $S_3 = \{7\} \neq \emptyset$. Además, $\{s \in S_{j-1} \mid s > r_j\} = \{s \in \{5, 7\}, s > 6\} = \{7\}$. Por ello, v' no puede ser prefijo del vector asociado a $\hat{C}'|_{n-2}$.

Luego eso nos fuerza a probar $\lambda = l_j + 1 = 6$. Mas $\left[0 \quad 3 \mid 1 \quad 2 \mid 4 \quad 7 \mid 6\right]$ no puede ser prefijo de ningún vector para ningún ρ ya que no se cumple el lema 2.4.7: $\hat{S}_{j-1} = S_{j-1} \setminus \{\lambda\} = \{5\}$ es no vacío y $6 > \text{mín}\, \hat{S}_{j-1} = 5$.

Prosigamos con el siguiente valor de j, $j = 2$. Sin embargo, ya que $r_j + 1 = 8$ y 8 no es un valor válido, no existe ρ que permita que $v' = \left[0 \quad 3 \mid 1 \quad 2 \mid 4 \quad \rho\right]$ sea prefijo de alguna clase $\hat{C}'|_2$ mayor que $\hat{C}|_2$. Por el lema 2.5.3, sabemos que el siguiente candidato es $\lambda = 5$. Sin embargo, se da que $\text{mín}\, S_{j-1} \setminus \{5\} = \text{mín}\{4, 6, 7\} = 4 < 5 = \lambda$, o lo que es lo mismo, no se cumple el lema 2.4.7.

Esto nos conduce al caso $j = 1$ y $\rho = r_j + 1 = 3$. Este caso es inviable ya que se viola el lema 2.4.5: $|S_{j-1} \setminus \{l_j, \rho\}| = |\{2, 4, 5, 6, 7\}| = 5 > 4 = h(v, v' + \rho)$, donde $v' = \left[0 \quad 3 \mid 1\right]$. Finalmente, ya que $\{s \in S_{j-1} \mid s > r_j\} = \{4, 5, 6, 7\} \neq \emptyset$, es obligado comprobar el teorema 2.4.8 con $\rho = 4$. Es rutinario comprobar que con este valor de ρ el teorema se satisface para v' y se deja a comprobar. □

Este ejemplo pone de manifiesto que la idea fundamental de este algoritmo es, de manera informal, *rellenar el primer hueco que se pueda rellenar*; así como exhibir que no hay grandes ideas en el mismo, solo el uso de pequeñas herramientas de manera sucesiva. Pero en realidad, el objetivo ha sido y sigue siendo completar el prefijo v' hasta obtener v'' de manera que efectivamente la diferencia de índices sea 1. Para ello, pongamos sobre la mesa algún lema más.

Lema 2.5.6 (Incremento retardado)**.** *Dado v' prefijo de tamaño $2p > 2(j+1), j > 0$ de v tal que $h(v, v') \geq 2$ y dados los circuitos $(l_{j-1}, r_{j-1}), (l_j, r_j)$ se da la siguiente alternativa:*

- *O bien existe un único circuito (λ, ρ) tal que el índice entre $t_j(l_j, r_j)$ y $\tau(\lambda, \rho)$ dista 1, donde τ y t_j vienen dados por las siguientes condiciones:*
  - *$\tau = 1$ si $\lambda = \rho$ o $\lambda = l_j, \rho = r_j$; y $\tau = 0$ en otro caso.*
  - *$t_j = 1$ si $l_j = r_j$ o $l_j = l_{j-1}, r_j = r_{j-1}$; y $t_j = 0$ en otro caso.*
- *O bien $l_j = r_j = k - 1$ y el índice de $1(l_j, r_j)$ es $k^2 - 1$.*

Demostración. Supongamos que no se da que $l_j = r_j \neq k-1$ (con lo que el índice de $1(l_j, r_j) \neq k^2 - 1$) y veamos que se concluye el resultado del lema. Si $l_j = r_j$, entonces tomemos $\lambda' = l_j, \rho' = r_j + 1$; si $l_j = l_{j-1}$ y $r_j = r_{j-1}$, tomemos $\lambda' = l_j, \rho' = r_j + 1$ y si no, tomemos $\lambda' = l_j, \rho' = r_j$. Así, si $\rho' = k$, tomemos $\rho = \lambda = \lambda' + 1$; y si no, elijamos $\lambda = \lambda'$, $\rho = \rho'$. Es rutinario con la fórmula del índice sobre $\hat{\mathcal{C}}_{n,w}|_d$ comprobar que son valores válidos y que dicha elección es la adecuada. □

Notemos que podemos extender este lema para el caso $j = 0, p > 1$ tomando $\lambda = l_0, \rho = r_0$ si

$l_0 \neq r_0$, $\lambda = l_0$ y $\rho = r_0 + 1$ si $r_0 < k$ y $l_0 = r_0$ y $\lambda = \rho = l_0 + 1$ en cualquier otro caso; con lo que usaremos el lema en su versión extendida.

Este lema debe su nombre a que podemos detectar de antemano si debemos o no incrementar el valor asociado a los cables l y r, y en caso de hacerlo, avisar de antemano si podemos o no retardar dicho incremento 1 ciclo (nótese que si $l \neq r$ tenemos los circuitos $0(l, r)$ y $1(l, r)$). Por otra parte y por simplicidad en la notación, denotaremos S'_j al conjunto de restantes tomando como vector complemento v'.

Lema 2.5.7 (Paso grande)**.** *Si $h(v, v') \geq |S'_j| + 2$ y v' es prefijo de tamaño $2j$ de un vector v'' cuyo índice dista 1 del de v, entonces $\hat{v}' = v' + \lambda + \rho$ es un prefijo asociado a v''; donde λ y ρ son el resultado del lema 2.5.6. Además, $h(v, \hat{v}') \geq |S'_j|$.*

Demostración. Por la elección de λ y ρ, sabemos que $\hat{f}_n(\tau(l_j, r_j)) + 1 = \hat{f}_n(\tau'(\lambda, \rho))$, donde $\tau = 1$ si $l_j = r_j$ o si $j > 0, l_j = l_{j-1}, r_j = r_{j-1}$ y 0 en otro caso, y $\tau' = 1$ si $\lambda = \rho$ y 0 en otro caso. Además, es rutinario mostrar que se cumple la propiedad del buen prefijo para $\hat{v}'$ teniendo en cuenta que $h(v, v') = h(v'', v')$ y utilizando v'' como vector base (y en consecuencia S'_j como conjunto de restantes). Finalmente, repasando con cuidado la definición del conjunto V en la demostración de 2.4.9 y la definición de λ y ρ en el lema 2.5.6, es inmediato comprobar que $v' + \lambda + \rho$ es prefijo también de v''. □

Lema 2.5.8 (Paso pequeño)**.** *Si $h(v, v') = |S'_j| + 1$ y v' es prefijo de tamaño $2j$ de un vector v'' cuyo índice dista 1 del de v, entonces existe un único vector $\hat{v}' = v' + \lambda + \sigma$ asociado a v''; donde $l_j \leq \lambda \leq l_j + 1$ y $\sigma \in S'_j$.*

Demostración. Sean λ', ρ' los resultantes del lema 2.5.6. Así, y dado que $S'_j \neq \emptyset$ por hipótesis, podemos contar con la siguiente casuística:

1. $\lambda' = l_j + 1$
2. $\lambda' = l_j$, $\rho' \leq \max S'_j$
3. $\lambda' = l_j$, $\rho' > \max S'_j$

En el primer caso, el conjunto $\{s \geq \lambda', s \in S'_j\} \neq \emptyset$ ya que si no para cualquier valor $\lambda'' \geq \lambda'$, $\{s \geq \lambda'', s \in S'_j\} = \emptyset$, con lo que no existirían λ, ρ de tal manera que a la vez se dé que $\rho \geq \lambda$ y $S'_{p-1} = \emptyset$. Y eso contradice las hipótesis ya que se presupone que v' es prefijo de v'' y v'' es un vector de alguna clase $\hat{C}'|_d$. Así, podemos tomar $\lambda = \lambda'$ y $\sigma = \min\{s \geq \lambda', s \in S'_j\} \geq \lambda$.

En el segundo caso, se da por hipótesis que $\{s \geq \rho', s \in S'_j\} \neq \emptyset$. Y dado que $\rho \leq \max S'_j$, entonces tomando $\lambda = \lambda'$ y $\sigma = \min\{s \geq \rho', s \in S'_j\}$, es claro que se da la siguiente desigualdad $\lambda \leq \rho' \leq \sigma$.

En el tercer y último caso, dado que $\{s \geq \rho', s \in S'_j\} = \emptyset$ tomemos $\lambda = \lambda' + 1$, $\sigma = \min S'_j$. Además, por el lema 2.4.7, es claro que $\lambda \leq \sigma$.

Finalmente, es rutinario comprobar que $\hat{v}' = v' + \lambda + \sigma$ cumple la propiedad del buen prefijo y es prefijo de v''; al igual que ver que $h(v, \hat{v}') \geq |S'_j \setminus \{\lambda, \sigma\}|$. □

Lema 2.5.9 (Avalancha)**.** *Si $h(v, v') = |S'_j|$ y v' es prefijo de tamaño $2j$ de un vector v'' cuyo índice dista 1 del de v, entonces $\hat{v}' = v' + \sigma + \sigma'$ es un prefijo asociado a v'', donde $\sigma = \min S'_j$ y $\sigma' = \min S'_j \setminus \{\sigma\}$. Además, $h(v, \hat{v}') = |S'_j| - 2 = |S'_j \setminus \{\sigma, \sigma'\}|$.*

Demostración. La prueba es análoga a la del lema 2.5.7 y no presenta ningún tipo de dificultad añadida. □

Estos tres últimos lemas combinados nos ayudarán a completar el prefijo v' obtenido en el algoritmo 2.5.4. Observemos el porqué intuitivo del nombre de cada uno de los lemas. En primer lugar, el lema Paso grande hace referencia a que dados los λ y ρ del lema 2.5.6, puede completar v' añadiendo los dos. Sin embargo, el lema Paso pequeño solo puede utilizar λ (o $\lambda + 1$) para completar v'; el valor ρ se difumina. Finalmente, el lema Avalancha fuerza a que una vez se dé ese lema, siempre se va a cumplir mientras que $v' \neq v''$, por eso es una avalancha: todos los valores de S_j se añaden uno tras otro sin pausa. Mostremos ahora cómo utilizar estos lemas para, dado un vector v, obtener un vector v'' tal que el índice entre v y v'' diste 1.

Algoritmo 2.5.10 (Incremento del cableado)**.** *Dado $v = (l_0, r_0, ..., l_{p-1}, r_{p-1})$ vector de $\hat{C}|_d$, obtenemos el vector v'' que diste 1 de v vía el índice si existe.*

Algoritmo 4 Incremento del cableado

Require: $v = (l_0, r_0, ..., l_{p-1}, r_{p-1})$.
  $v' \leftarrow$ Prefijo mínimo
  if v' es *None* **then**
    return *None* ▷ Si no existe v', no se puede extender.
  end if
  $S \leftarrow S'_j$ ▷ El algoritmo Prefijo mínimo puede devolver este conjunto también.
  $j \leftarrow \frac{|v'|}{2}$

while $j < p$ **do**
  if $h(v, v') \geq |S| + 2$ **then**
    Obtener λ y ρ del lema Incremento retardado.
    $v' \leftarrow v' + \lambda + \rho$ ▷ Estamos en las hipótesis del lema Paso grande.
    $S \leftarrow S \setminus \{\lambda, \rho\}$
  else if $h(v, v') = |S| + 1$ **then**
    Obtener λ y σ mediante el lema Paso pequeño.
    $v' \leftarrow v' + \lambda + \sigma$
    $S \leftarrow S \setminus \{\lambda, \sigma\}$
  else
    Obtener σ y σ' mediante el lema Avalancha.
    $v' \leftarrow v' + \sigma + \sigma'$
    $S \leftarrow S \setminus \{\sigma, \sigma'\}$
  end if
  $j \leftarrow j + 1$
end while
return $v'' = v'$

Demostración. Se sigue de los lemas 2.5.6, 2.5.7, 2.5.8 y 2.5.9. □

Notemos que manteniendo de nuevo S como un montículo, el coste de calcular los valores λ, ρ con los que extender v' está en $\mathcal{O}(\log(p))$. Por tanto, de manera global el coste de este algoritmo está en $\mathcal{O}(p \log(p))$. Además, la tabla de ocurrencias se puede ir actualizando a medida que extendemos v' en tiempo constante, de manera que la podamos reutilizar en llamadas sucesivas a este algoritmo. Veamos con un ejemplo el funcionamiento de este algoritmo.

Ejemplo 2.5.11. Supongamos que contamos con la representación matricial dispersa de la clase $\hat{\mathcal{C}}|_{n-2}$ (supuesto $n > 2$).

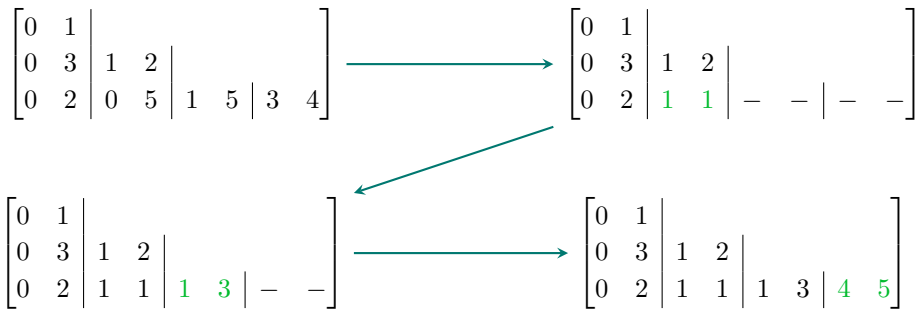


Figura 2.8: Ejecución del algoritmo 2.5.10 a partir de $\hat{C}|_{n-2}$.

Calculemos la clase $\hat{C}'|_{n-2}$ que dista 1 de $\hat{C}|_{n-2}$. En primer lugar, notemos que el prefijo

mínimo de $\left[0 \quad 2 \mid 0 \quad 5 \mid 1 \quad 5 \mid 3 \quad 4\right]$ es $v' = \left[0 \quad 2 \mid 1\right]$ (compruébese). Por el corolario 2.4.10 y el teorema 2.4.8, sabemos que cualquier ρ del cual $v' + \rho$ sea prefijo cumple que $r_j = 1$. Por ello, v'' tiene de prefijo a $v' = \left[0 \quad 2 \mid 1 \quad 1\right]$.

Entonces, dado que $h(v'', v') = 4 = |S'_1| + 1$, estamos en las hipótesis del lema 2.5.8. Los valores obtenidos por el lema 2.5.6 son $\lambda' = 1, \rho' = 2$, con lo que estamos en el subcaso 2 de dicho lema. Por ello, sabemos que v'' tiene por prefijo $v' + \lambda + \sigma$, donde $\lambda = \lambda' = 1$ y $\sigma = \text{mín}\{s \geq \rho', s \in S'_j\} = \text{mín}\{3, 4, 5\} = 3$.

En la siguiente etapa del algoritmo sabemos que $v' = \left[0 \quad 2 \mid 1 \quad 1 \mid 1 \quad 3\right]$ es prefijo de v''. Además, $h(v'', v') = 2 = |S_j|$, con lo que estamos en las hipótesis del lema 2.5.9. Así, es claro que $v'' = v' + 4 + 5$ ya que $h(v'', v' + 4 + 5) = 0$. □

Ejemplo 2.5.12. Compruébese que el vector v'' del cual es prefijo el vector v' del ejemplo 2.5.5 es $v'' = \left[0 \quad 3 \mid 1 \quad 4 \mid 2 \quad 5 \mid 6 \quad 7\right]$ (lo cual es inmediato al cumplirse el lema 2.5.9 con v'). □

Además, observemos que no hemos hablado de variar los inputs. Es decir, estos lemas suponen que $d \geq 1$ y en definitiva, como mostraremos, nos permitirá obtener un circuito bien definido sobre un conjunto de $k = |E_0(C)| \leq m$ inputs. Sin embargo, no hemos hablado de cómo seleccionar esta cantidad. La respuesta es muy simple: dado un subconjunto S de $\{i \mid 0 \leq i < m\}$ tal que $|S| = k$, podemos ordenar los elementos de dicho conjunto vía el índice. Los elementos presentes en S determinan unívocamente qué inputs poder elegir y, ya que esto no presenta ningún tipo de complicación técnica, podremos suponer que contamos con un método que nos permite cambiar de inputs sin cambiar la estructura del circuito.

Finalmente, notemos que tras obtener v'' mediante este algoritmo, podemos aplicar el corolario 2.3.2 para obtener el mínimo vector g asociado a $\hat{C}'|_d$ y así definir la clase $C'|_d \in \mathcal{C}_{n,w}|_d$. Además notemos que en todo el proceso realizado no hemos perdido información, esto es, dado $C|_d$ podemos obtener $C'|_d$ de manera que todos los niveles no afectados por la teoría mencionada anteriormente no cambian. Por ello, podemos visualizar $\hat{C}|_d$ como una abstracción para simplificar los cálculos de la cual podemos olvidarnos de manera práctica.

Recapitulando ideas: hemos conseguido un algoritmo que nos permite, dada una clase $\hat{C}|_d$, detectar si hay alguna clase $\hat{C}'|_d$ siguiente a $\hat{C}|_d$ de manera que $\hat{C}|_{d+1} = \hat{C}'|_{d+1}$ (gracias a la propiedad del buen prefijo); y en caso afirmativo, construirla.

2.6. Ampliación del suelo, cambio de nivel y fin del algoritmo

Sin embargo, podría suceder que la clase $\hat{C}'|_d$ que dista 1 de $\hat{C}|_d$, donde $\hat{C}|_{d+1} = \hat{C}'|_{d+1}$ dentro de $\hat{\mathcal{C}}_{n,w}|_{d+1}$ no cumpliese que $|E_{d-1}(\hat{C}')| = |E_{d-1}(\hat{C})|$. En ese caso, tendría que darse que $|E_{d-1}(\hat{C}')| = |E_{d-1}(\hat{C})| + 1$ ya que si existe $j, d \leq j \leq n$ tal que$|E_j(\hat{C})| \neq |E_j(\hat{C}')|$, entonces $\hat{C}|_{d+1} \neq \hat{C}'|_{d+1}$. Por el lema 2.1.4, el corolario 1.3.3 y la definición de $\hat{\mathcal{C}}_{n,w}$, sabemos que esto solo se puede dar si $|E_{d-1}(\hat{C})| < \text{mín}\{2p, w, \delta_m(d)\}$.

Lema 2.6.1 (Vector mínimo)**.** *Dado $d \geq 1$ y una clase $\hat{C}|_{d+1} \in \hat{\mathcal{C}}_{n,w}|_{d+1}$ con $n > 1$ tal que $p = |E_d(\hat{C})|$ y $k = |E_{d-1}(\hat{C})|$, entonces existe un único vector v_{d-1} (asociado a una única clase $\hat{C}'|_d$) de manera que $\hat{C}'|_d$ sea mínima dentro de $\hat{C}|_{d+1}$.*

Demostración. Por la definición de bien conectado y la función índice es claro que (0) es prefijo de cualquier vector v. Así, gracias al corolario 2.4.10 existe un ρ tal que $v' = (0, \rho)$ es el mínimo prefijo de tamaño 2. En concreto, este vector es prefijo del mínimo vector de v asociado a alguna clase $\hat{C}|_d$. Aplicando el algoritmo 2.5.10 podemos extender v' hasta v. La unicidad se da del determinismo del algoritmo. □

Teorema 2.6.2 (Ampliación del suelo)**.** *Existe una clase $\hat{C}'|_d$ de índice 1 más que el de $\hat{C}|_d$ con $|E_{d-1}(\hat{C}')| = k + 1$ cumpliendo que $\hat{C}|_{d+1} = \hat{C}'|_{d+1}$ si y solo si $k < \min\{2p, w, \delta_m(d-1)\}$ y el algoritmo 2.5.10 devuelve None para $\hat{C}|_d$.*

Demostración.
$\Longrightarrow$ Sea $\hat{C}'|_d$ tal que $|E_d(\hat{C}')| = k + 1$. Por el lema 2.1.4 se tiene que $k + 1 \leq 2p$ y por el corolario 1.3.3 se da que $k + 1 \leq \delta_m(d-1)$. Además, por la definición de $\hat{\mathcal{C}}_{n,w}$ es claro que $k + 1 \leq w$. En definitiva, $k + 1 \leq \min\{w, \delta_m(d), 2p\}$. Por tanto, $k < \min\{w, \delta_m(d-1), 2p\}$. Finalmente, el algoritmo 2.5.10 debe devolver *None* ya que el índice entre $\hat{C}'|_d$ y $\hat{C}|_d$ dista 1 y no se cumple que $|E_{d-1}(\hat{C})| = |E_{d-1}(\hat{C}')|$.

$\Longleftarrow$ Si el algoritmo 2.5.10 devuelve *None* para $\hat{C}|_d$ significa que la siguiente clase $\hat{C}'|_d$, de existir, cumple que $|E_{d-1}(\hat{C}')| > k$. Como $k < \min\{w, \delta_m(d-1), 2p\}$, tomemos $k' = k + 1$ y comprobemos que existe una clase $\hat{C}'|_d$ tal que $|E_{d-1}(\hat{C}')| = k'$, ya que si $k'' > k'$, la clase $\hat{C}''|_d$ dentro de $\hat{C}|_{d+1}$ tendrá mayor índice.

De hecho, nos basta probar que existe una clase $\hat{C}'|_d \in \hat{C}|_{d+1}$ tal que $|E_{d-1}(\hat{C}')| = k'$, ya que por el lema 2.6.1, entonces dicha clase mínima existiría. Así, tomemos $v' = (0)$ y comprobemos que se cumple la propiedad del buen prefijo para algún v''. Tomando $\rho = 1$, puesto que $|S'_{-1} \setminus \{0, 1\}| = k' - 2 = k - 1 = |S_{-1}| - 1$, entonces $|S'_{-1} \setminus \{0, 0\}| = |S_{-1}| - 1 \leq 2(p-1) = h(v'', v' + \rho)$ para cualquier v'' de tamaño $2p$. Con lo que se cumple el lema 2.4.5.

Por otra parte, ya que $\hat{C}|_d$ es una clase, entonces el vector $\tilde{v}' = (0)$ es prefijo de su vector $\tilde{v}$ asociado y se cumple el lema 2.4.6. Es decir, $2(p-1) = h(\tilde{v}, \tilde{v}' + r_0) \leq 2(k^2 - 1 - \hat{f}_n(\tau(0, r_0))) \leq 2(k^2 - 1)$. Comprobemos que dicho lema se cumple también para v': $2(k'^2 - 1 - \hat{f}_n(1(0,0))) = 2(k^2 + 2k + 1 - 1 - 0) = 2(k^2 + 2) \geq 2(k^2 - 1) \geq 2(p-1)$. Por ello, v' cumple el teorema 2.4.8 y en consecuencia existe un vector v'' asociado a $\hat{C}'|_d$. Finalmente, nos queda comprobar que dicha clase está en $\hat{\mathcal{C}}_{n,w}$ y es fruto de un circuito con m inputs, cosa que en el resto de lemas anteriores era algo claro y evidente ya que el número de puertas en la clase $\hat{C}|_d$ era invariante a las transformaciones realizadas.

Para probarlo, probemos en primer lugar que pertenece a $\hat{\mathcal{C}}_{n,w}|_d$. Para ello, hay que probar que existe $\hat{D}$ en $\hat{\mathcal{C}}_{n,w}$ tal que $\hat{D}|_d = \hat{C}'|_d$. Puesto que $C \in \mathcal{C}_{n,w}$, entonces existe un valor l, tal que

$|E_l(C)| = w$, y este no puede ser d por hipótesis. Así, si existe un $l > d$ tal que $|E_l(C)| = w$, entonces $|E_l(C')| = w$ y hemos acabado. Si no, necesariamente existe un $l < d$ tal que $|E_l(C)| = w$. Probaremos que existe una clase $\hat{C}|_l \in \hat{C}|_d$, con $|E_l(C)| = w$ y lo haremos de manera constructiva, obteniendo para cada $j, l < j < d$ un vector v_j de manera que dichos definan la clase $\hat{C}|_d$.

Por la proposición 2.1.4 sabemos que si en el nivel j hay k_j puertas, en el nivel $j+1$ hay al menos $\eta(k_j)$ puertas. Así, definamos $k_j = \text{máx}\{k+1, \eta^{j-l}(w)\}, l \leq j \leq d$. Ya que $k < k+1$ y C es un circuito se da necesariamente que $\eta^{d-l}(w) \leq k < k+1$, con lo que $k_d = k+1$. Además, se cumple que $\eta^{l-l}(w) = id(w) = w$, lo que da que $k_l = w$ y de manera obvia obtenemos que $k_j \leq k_{j-1}$. Entonces se da la siguiente causística para todo valor $j,\ l < j < d$:

- $k_j = k_{j-1} = k+1$. Tomemos en esta situación el vector $v_j = (0, 0, 1, 1, ..., k, k)$. Entonces, podemos definir la clase $\hat{C'}|_j$ como $\hat{C'}|_{j+1} \cup \{v_j\}$. Además, se cumple que $|E_j(\hat{C'}|_j)| = |E_j(\hat{C'}|_{j+1})| = k_j$ y $|E_{j-1}(\hat{C'}|_j)| = k_{j-1} = k_j$.
- $k_j = k+1, k_{j-1} = \eta^{j-1-l}(w) > k+1$. Por la proposición 2.1.4 sabemos que se da la siguiente desigualdad $k_j < k_{j-1} \leq 2k_j$. Así, denotemos $p = 2k_j - k_{j-1} > 0$ y definamos el vector $v_{j-1} = (0, 0, 1, 1..., p-1, p-1, p, p+1, ..., k_{j-1})$. Es claro que es un vector de tamaño $2k_j$ y se cumple que aparecen todos los elementos entre 0 y k_{j-1}, con lo que la clase $\hat{C'}|_j = \hat{C'}|_{j+1} \cup \{v_j\}$ está bien definida y cumple que $|E_j(\hat{C'}|_j)| = |E_j(\hat{C'}|_{j+1})| = k_j$ y $|E_{j-1}(\hat{C'}|_j)| = k_{j-1}$.
- $k_j = \eta^{j-l}(w) > k+1, k_{j-1} = \eta^{j-1-l}(w)$. En este caso o bien $2k_j = k_{j-1}$, con lo que definamos $v_j = (0, 1, 2, ..., k_{j-1})$ o bien $2k_j - 1 = k_{j-1}$, en cuyo caso definamos $v_j = (0, 0, 1, 2, ..., k_{j-1})$. En ambos casos v_j es un vector de tamaño $2k_j$ y podemos definir la clase $\hat{C'}|_j$ como $\hat{C'}|_{j+1} \cup \{v_j\}$, de manera que $|E_j(\hat{C'}|_j)| = |E_j(\hat{C'}|_{j+1})| = k_j$ y $|E_{j-1}(\hat{C'}|_j)| = k_{j-1}$.

En cualquier caso, es claro que podemos obtener la clase $\hat{C'}|_{l+1}$ a partir de $\hat{C'}|_d$ cumpliendo que $|E_l(\hat{C'})| = w$ y $\hat{C'}|_{l+1} \in \hat{C'}|_d$. Por tanto, $\hat{C'}|_d \in \hat{\mathcal{C}}_{n,w}|_d$. Además, ya que $|E_l(\hat{C'}|_{l+1})| = |E_l(C)| = w$, es claro que $w \leq \delta_m(l-1)$. Queda por probar pues que hay un circuito en esa clase generable mediante m inputs.

Definamos $k_l = |E_l(\hat{C'})|$ (con $l = d-1$ en el caso de que $\hat{C'}|_d$ tuviese un nivel $i > d$ tal que $|E_i(D)| = w$); y para todo $j,\ 0 \leq j < l$ escribamos $k_j = \sigma(k_{j+1})$. Así, probemos que si tenemos definido $\hat{C'}|_{j+1}$, podemos definir $\hat{C'}|_j$ de manera que $|E_{j-1}(\hat{C'}|_j)| = k_{j-1}$ y $k_{j-1} \leq \delta_m(j-1)$. De hecho, nos encargaremos de dar un vector v_j con el que definir dicha clase. Tomemos $v' = (0, 0)$. Por el lema 2.5.6 existen l_{i+1}, r_{i+1} de manera que dado el vector $(l_0, r_0, ...l_i, r_i)$ se dé que $f_j(\tau(l_i, r_i)) + 1 = f_j(\tau'(l_{i+1}, r_{i+1}))$, donde $\tau = 1$ si $l_i = r_i$ o $i > 0, l_{i-1} = l_i, r_{i-1} = r_i$ y 0 en otro caso y $\tau' = 1$ si $l_{i+1} = r_{i+1}$ o si $l_i = l_{i+1}, r_i = r_{i+1}$ y 0 en otro caso. Así, y ya que $l_0 = r_0 = 0$ y $f_j(1(0, 0)) = 0$ podemos definir un vector de tamaño $2k_j$ en el que aparezcan todos los términos $i,\ 0 \leq i < k_{j-1}$ ya que se da que $(k_{j-1} - 1)^2 < k_j \leq k_{j-1}^2$ (proposición 2.1.4).

Finalmente, ya que $k_{j-1}^2 \leq k_j \leq \delta_m(j)$, entonces $k_{j-1}^2 \leq \sqrt{\delta_m(j)} = \sqrt{m^{2^j}} = m^{2^{j-1}} = \delta_m(j-1)$. Repitiendo el argumento un número finito de veces obtenemos que hemos definido una clase $\hat{C'}|_0$ de manera que $k_0 \leq \delta_m(0) = m$. Por tanto, la clase $\hat{C'}|_{l+1}$ pertenece a $\hat{\mathcal{C}}_{n,w}|_l$ y en consecuencia esto

mismo sucede para $\hat{C}'|_d$.

□

Sin embargo, podría darse que existiera una clase $\hat{C}'|_d$ de índice mayor que el de $\hat{C}|_d$ que no cumpliese que $\hat{C}|_{d+1} = \hat{C}'|_{d+1}$. En ese caso es porque el índice de $\hat{C}|_d$ inducido en $\hat{C}|_{d+1}$ es máximo; es decir $\hat{C}|_d$ es la última clase de $\hat{C}|_{d+1}$. En este caso, sabemos que dicha clase tiene que cumplir que $\hat{C}'|_{d+1}$ diste 1 de $\hat{C}|_{d+1}$ y de manera recursiva podemos aplicar las herramientas que hemos desarrollado en esta sección para obtener una clase $\hat{C}'|_j$ tal que $\hat{C}|_j$ diste 1.

Combinando estos dos lemas junto al algoritmo 2.5.10 podremos avanzar dentro de $\hat{C}|_{d+1}$ de manera cómoda y natural, tal y como se puede ver en el ejemplo siguiente.

Ejemplo 2.6.3. Sea $\hat{\mathcal{C}}|_{n-3} \in \mathcal{C}_{n,w}$ la clase con representación matricial dispersa siguiente (supuesto $n > 3, w \geq 6$ y $\delta_m(n-4) \geq 6$):

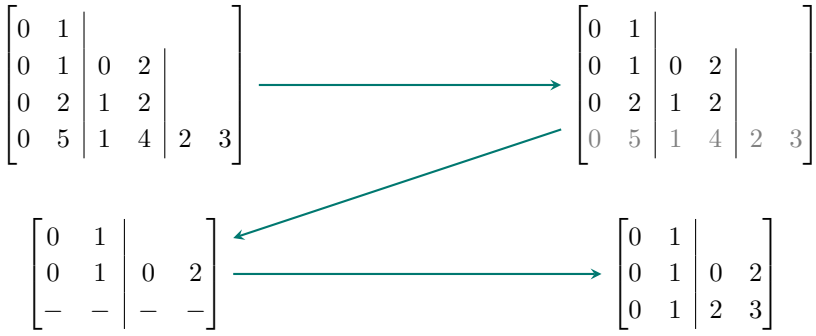


Figura 2.9: Ejemplo de cambio de nivel.

Observemos que la clase $\hat{C}|_{n-3}$ cumple que $|E_{n-4}(\hat{C})| = 6$ ya que los valores distintos de la última fila son lo que determina dicho valor. En estas condiciones, el algoritmo 2.5.10 devuelve *None* y el teorema 2.6.2 no se cumple ya que $k = \text{mín}\{6, w, \delta_m(n-4)\}$. Por tanto, nos olvidamos de $\hat{C}|_{n-3}$ y pasamos a considerar $\hat{C}|_{n-2}$, de lo que da fe el color grisáceo que toma la fila $n-3$.

En este caso, el algoritmo 2.5.10 vuelve a dar *None*, pero esta vez sí se cumple el teorema 2.6.2 ya que $3 < \text{mín}\{2 \cdot 2, w, \delta_m(n-3)\} = 4$. Por tanto sabemos que la nueva clase $\hat{C}'|_{n-2}$ cumple que $|E_{n-2}(C')| = 2, |E_{n-3}(C')| = 4$. Utilizando el lema 2.6.1, podemos fácilmente obtener que $v_{n-3} = \left[\begin{array}{cc|cc} 0 & 1 & 2 & 3 \end{array}\right]$. □

Como se ha podido ver, a veces es necesario comprobar de manera sucesiva dicho lema para poder obtener la siguiente clase. Además, en cada subida, perdemos información sobre la estructura del resto de niveles, información que de alguna manera tendremos que recuperar de nuevo.

Corolario 2.6.4 (Fin del algoritmo)**.** *Si* $|E_{n-1}(\hat{C})| = \text{mín}\{2p, w, \delta_m(n-1)\}$ *y no existe* $\hat{C}'|_{n-1}$ *tal que la diferencia de índices con* $\hat{C}|_{n-1}$ *diste* 1*, entonces no existe ningún circuito* $\hat{C}'$ *de índice*

mayor que $\hat{C}$ *en* $\hat{\mathcal{C}}_{n,w}$.

Demostración. Se deduce trivialmente del teorema 2.6.2 con $d = n$. □

2.7. Retorno a los circuitos: representante mínimo y circuito inicial

Todos los resultados anteriores nos ayudan a avanzar en $\mathcal{C}_{n,w}$ con la abstracción de las clases. Sin embargo, el objetivo inicial era trabajar con circuitos, no con clases. Así, trataremos de, una vez obtenida una clase, recuperar el mínimo circuito de dicha clase. En primer lugar, notemos que todo circuito C es equivalente a $C|_0$. Por ello, nos basta con tener un método para descender entre clases de manera que dado $C|_d$ el representante $C|_{d-1}$ sea mínimo dentro de $C|_d$. De hecho, nos basta con analizarlo desde $\hat{\mathcal{C}}_{n,w}|_d$ ya que podemos aplicar el corolario 2.3.2 entre cada representante $\hat{C}|_d$ para obtener la minimalidad (es rutinario comprobarlo utilizando el índice).

Lema 2.7.1. *Sea* C *un circuito de profundidad* $n > 1$ *y anchura arbitraria y sean* $d, d' \geq 0, d > d'$, $k = |E_d(C)|, k' = |E_{d'}(C)|$ *tal que* $\eta^{d-d'}(k') \leq k$ *y* $\sigma^{d-d'}(k) \leq k'$. *Entonces existe una clase* $\hat{C}' \in \hat{\mathcal{C}}_n|_{d'}$ *que cumple que* $\hat{C}'|_{d+1} = \hat{C}|_{d+1}$ *y que para todo* $l, d' \leq l \leq d$ *se da que* $|E_l(\hat{C}')| = k_l$, *donde* $k_l = \max\{\sigma^{d-l}(k), \eta^{l-d'}(k')\}$. *Además el índice de esta clase (obtenido al comparar clases de* $\mathcal{C}_n|_{d'}$ *dentro de* $\hat{C}|_{d+1}$*) es mínimo.*

Demostración. En primer lugar notemos $\sigma(z) \leq z$ y $\eta(z) \leq z$ para cualquier $z \geq 0$ y que $k_d = k$ y $k'_d = k'$. Gracias al corolario 2.1.5 sabemos que $|E_l(C)| \geq \sigma^{d-l}(k)$. Por otra parte, aplicando la proposición 2.1.4 de manera sucesiva, obtenemos que $|E_l(C)| \geq \eta^{l-d'}(k')$. Por ello, $|E_l(C)| \geq \max\{\sigma^{d-l}(k), \eta^{l-d'}(k')\}$. Además, del hecho de que C sea un circuito, es claro que podemos tomar el resto de niveles de C' no mencionados en el lema idénticos a C. Finalmente, gracias de nuevo a la proposición 2.1.4, es claro que una clase que cumpla que $|E_l(\hat{C})| = k_l$ y $|E_{l-1}(\hat{C})| = k_{l-1}$ está bien definida.

El *además* es claro gracias a la aplicación reiterada del lema 2.6.1 para todo $l \neq 0$ y del de menor subconjunto si $l = 0$; teniendo en cuenta que todos los resultados mencionados se aplican independientemente de la anchura del circuito (aunque las usemos siempre con restricción). □

Corolario 2.7.2 (Extensión mínima)**.** *Sea* C *un circuito en* $\mathcal{C}_{n,w}$ *y sean* $d, d' \geq 0, d > d'$, $k = |E_d(C)|, k' = |E_{d'}(C)|$ *tal que* $\eta^{d-d'}(k') \leq k$ *y* $\sigma^{d-d'}(k) \leq k'$ *y para algún nivel* d'' *de la forma* $d'' \geq d$ *o* $d'' \leq d'$ *se da que* $|E_{d''}(C)| = w$. *Entonces existe una clase* $\hat{C}' \in \mathcal{C}_{n,w}|_{d'}$ *que cumple que* $\hat{C}'|_{d+1} = \hat{C}|_{d+1}$ *y que para todo* $l, d' \leq l \leq d$, *se da que* $|E_l(\hat{C}')| = k_l$, *donde* $k_l = \max\{\sigma^{d-l}(k), \eta^{l-d'}(k')\}$. *Además el índice de esta clase (obtenido al comparar clases de* $\mathcal{C}_{n,w}|_{d'}$ *dentro de* $\hat{C}|_{d+1}$*) es mínimo.*

Demostración. Por el lema 2.7.1 sabemos que $k_l \geq \text{máx}\{\sigma^{d-l}(k), \eta^{l-d'}(k')\}$ y por la observación realizada en dicho lema, $k_l \leq \text{máx}\{k, k'\}$. Así, del hecho de que para todo $l, d' < l < d$, se dé que $l \neq d''$, entonces es claro que la clase $\hat{C'}|_d$ definida en dicho lema está en $\hat{\mathcal{C}}_{n,w}|_d$. □

Teorema 2.7.3 (Representante mínimo)**.** *Sea $d \geq 0$, y sea una clase $\hat{C}|_{d+1}$ de manera que $k = |E_d(\hat{C})|$. Entonces existe una clase $\hat{C'}|_0$ tal que $\hat{C'}|_{d+1} = \hat{C'}|_{d+1}$ de manera el índice de $\hat{C'}|_0$ sea mínimo.*

Demostración. En estas condiciones, podemos tener dos situaciones: que exista un nivel $l, d \leq l \leq n$, tal que $|E_l(C)| = w$ o que no exista dicho nivel.

Si existe dicho nivel, entonces nos es igual cuántos inputs tomar o el número de puertas empleadas en cada nivel siempre que el circuito esté bien formado (es decir, cumpla la proposición 2.1.4 y no tenga un nivel con más de w puertas). Así, dado que $\sigma^d(k) \leq \sigma^d(k)$ y $(\eta^d \circ \sigma^d)(k) \leq k$, podemos tomar $d' = 0, k' = \sigma^d(k)$ en el corolario 2.7.2 para obtener la mínima clase $\hat{C'}|_0$, ya que no hay menor valor de k' posible.

Si no, apliquemos el lema 2.1.7 para obtener el mínimo l tal que $|E_d(C')| = w$ para cualquier circuito C'. Ya que $l < d$, podemos aplicar el corolario 2.7.2 con $d = l$, $k = w$, $d' = 0$ y $k' = \sigma^l(w)$ para obtener una clase $\hat{C''}|_0$ tal que $\hat{C''}|_l = \hat{C}|_l$ (y en consecuencia $\hat{C''}|_d = \hat{C}|_d$) de manera que el índice de $\hat{C''}|_0$ sea mínimo dentro de $\hat{C}|_d$. Ahora bien, $\hat{C''}|_0$ se puede extender a un circuito tal y como observamos aquí. Con lo que podemos hablar de C'' o de $\hat{C''}|_0$ casi indistintamente (nos es igual qué $C'' \in \hat{C''}|_0$ tomar para continuar con la demostración).

Por tanto, apliquemos de nuevo el corolario 2.7.2 vía C'' con d y k definidos en las hipótesis y $d' = l$, $k' = w$; aprovechando el hecho de que la desigualdad $\eta^{d-l}(w) \leq k$ se da en el circuito C. Así, obtenemos una clase $\hat{C'}|_l$ tal que $\hat{C'''}|_d = \hat{C''}|_d = \hat{C}|_d$ de índice mínimo sobre $\hat{C}|_d$. Así, sea C''' un circuito de $\hat{C'''}|_l$. Apliquemos de nuevo el mismo corolario con C''', $d = l, k = w, d' = 0$ y $k' = \sigma^l(w)$ para obtener $\hat{C'}|_0$ de índice mínimo sobre $\hat{C'''}|_l = \hat{C}|_l$ (y en consecuencia de índice mínimo sobre $\hat{C}|_d$).

Notemos finalmente que por la demostración del lema 2.7.1, para todo j tal que $0 \leq j < l$, los vectores v_j asociados al nivel j-ésimo de $\hat{C'}|_0$ coinciden con los vectores v_j de $\hat{C''}|_0$, con lo que computacionalmente solo es necesario aplicar el corolario dos veces. □

El nombre de este teorema claramente tiene su origen en la posibilidad de partir de una clase de $\mathcal{C}_{n,w}|_d$ y transformarla en un circuito en $\mathcal{C}_{n,w}$ con mínimo índice.

Algoritmo 2.7.4 (Representante mínimo)**.** *Obtención del mínimo representante de $C|_d$*

Algoritmo 5 Representante mínimo

Require: d, C, v_p
  while $d > 0$ **do** ▷ Subrutina *extensión*.
    $p \leftarrow v_p[d]$
    $k \leftarrow v[d-1]$
    $C \leftarrow$Vector mínimo
    $g \leftarrow$ Tipo mínimo
    $C \leftarrow C \cup (v, g)$
    $d \leftarrow d - 1$
  end while
  return C

Require: d, $C|_d$
  $C \leftarrow C|_d$
  if Existe $j \geq d$ tal que $|E_j(C)| = w$ **then** ▷ Caso 1 del Representante mínimo
    $v_p \leftarrow$ Extensión mínima ▷ v_p contiene a todos los k_l.
  else ▷ Caso 2 del Representante mínimo
    $l \leftarrow$ Mínimo nivel
    $v_{p1} \leftarrow$ Extensión mínima ▷ Aplicado para obtener C''.
    $v_{p2} \leftarrow$ Extensión mínima ▷ Aplicado para obtener C'''.
    $v_p \leftarrow v_{p2} \cup v_{p1}$
  end if
  $C \leftarrow$ Subrutina *extensión*
  return C

Por las observaciones realizadas en el algoritmo 2.5.10, es claro que el coste de este algoritmo está en $\mathcal{O}(dp\log(p))$. Sin embargo, notemos que gracias a nuestro recorrido en dos etapas (tipos y cableado), conseguimos retardar la ejecución de este algoritmo dentro del nivel d.

Ejemplo 2.7.5. Veamos de manera práctica la aplicación del algoritmo 2.7.4 a la clase $\hat{C}|_3 \in \mathcal{C}_{4,4}$ sobre $m \geq 2$ inputs siguiente:

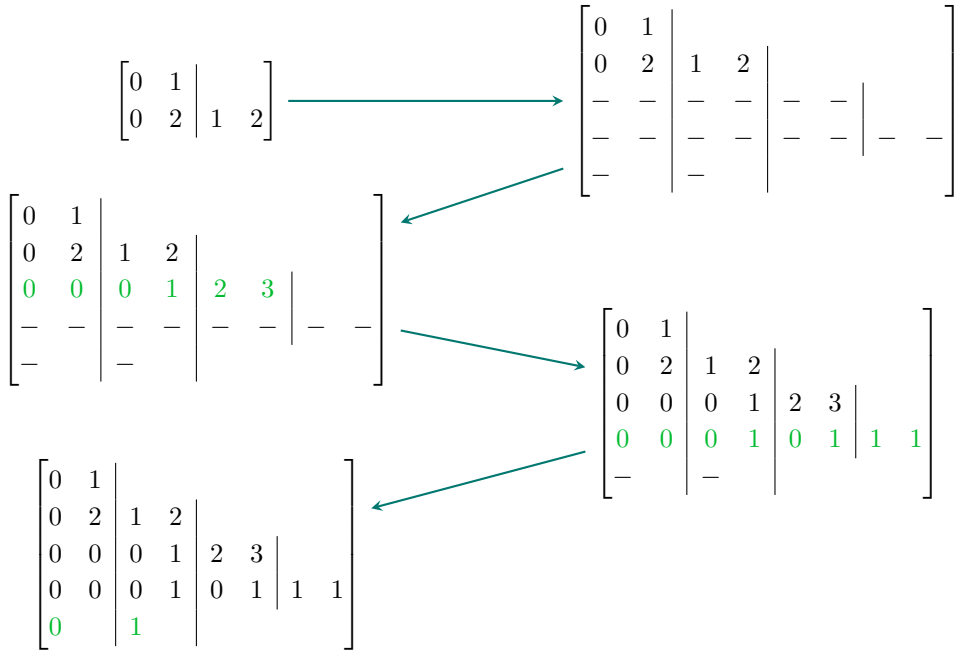


Figura 2.10: Ejecución del algoritmo 2.7.4 a partir de $\hat{C}|_3$.

En primer lugar, observemos que $\hat{C}|_2$ no tiene ningún nivel con 4 puertas. Así, aplicando el lema 2.1.7 obtenemos que $l = 2$ y por el corolario 2.7.2 obtenemos que $k_2 = 3, k_1 = 4, k_0 = 2$.

Así, el vector mínimo cuando $p = 3, k = 6$ es $\left[\begin{array}{cc|cc|cc} 0 & 0 & 0 & 1 & 2 & 3 \end{array}\right]$, el vector mínimo cuando $p = 6, k = 2$ es $\left[\begin{array}{cc|cc|cc|cc} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$ y finalmente, la mínima colección de 2 inputs es $\left[\begin{array}{c|c} 0 & 1 \end{array}\right]$ □

Ahora, y gracias a todos los lemas desarrollados podemos responder algo fundamental: ¿cuál es la primera clase que debemos tomar? Para ello, demostremos el siguiente teorema de manera constructiva.

Teorema 2.1.8. *El conjunto $\mathcal{C}_{n,w}$ no es vacío si y solo si dado $d = \lceil \log_2(\text{máx}\{2, \log_m(w)\}) \rceil$, se da que $\eta^{n-d}(w) = 1$.*

Demostración. Una de las implicaciones ya la demostramos en su momento, así que es momento de completar la prueba. En primer lugar observemos que existe una correspondencia natural entre los circuitos C de profundidad n y los circuitos de profundidad $n+1$ que son de la forma $C' \coloneqq id(C, C)$. La idea de esta demostración consiste en tomar un elemento arbitrario de $\mathcal{C}_n$ y obtener un elemento de $\mathcal{C}_{n,w}$.

Dado que $\eta^{n-d}(w) = 1$, obtenemos trivialmente que $\eta^{n-d}(w) \leq 1$ y $\sigma^{n-d}(1) = 1 \leq w$. Tomemos

$\hat{C}|_{n+1} = (0,0)$. Notemos que podemos replicar la demostración de la segunda parte del teorema 2.7.3 para $d = n, k = 1$ utilizando el lema 2.7.1 (sin garantizar en ningún momento que pertenece el circuito a $\hat{\mathcal{C}}_{n,w}$). Así, analizando la clase obtenida $\hat{C}|_0$, observamos que $|E_l(C)| = w$ y que $E_j(C) \leq w$ si $j \neq l$, con l obtenido vía el lema 2.1.7. Solo hemos tenido que cambiar la justificación de por qué $\eta^{n-d}(w) \leq 1$ para de manera constructiva obtener un elemento de $\hat{\mathcal{C}}_{n,w}$.

Finalmente, gracias al corolario 2.3.2 podemos obtener un representante $C'' \in C''|_{n+1}$. Este elemento está en $\mathcal{C}|_{n,w}$, con lo que hemos acabado la demostración. □

Además, notemos que el circuito obtenido es el mínimo circuito según el índice y a este circuito lo denominaremos *circuito inicial*. Como era de esperar, no podemos hablar del caso base de un algoritmo sin conocer su estructura. Por ello es su tardía aparición. Si repasamos con cuidado el ejemplo 2.1.13, es claro que este ejemplo representa el circuito inicial con los valores $m = 6, n = 4, w = 3$. Además, en este ejemplo se ve bien la necesidad de permutar los inputs, ya que este circuito utiliza solo 2 de los 6 que tiene a su disposición.

Algoritmo 2.7.6 (Circuito inicial)**.** *Obtención del circuito inicial.*

Algoritmo 6 Circuito inicial

Require: $n, w, m \in \mathbb{Z}, l = \lceil \log_2(\max\{2, \log_m(w)\}) \rceil, \eta^{n-l}(w) = 1$
  $C \leftarrow \emptyset$
  $C \leftarrow$ Representante mínimo
  return C

Demostración. La corrección de este algoritmo se sigue del teorema 2.1.8. Además, su coste está en $\mathcal{O}(nw \log(w))$, más solo se va a llamar una única vez. □

Ejemplo 2.7.7. Mostraremos, por completitud, un ejemplo de circuito inicial.

$$\left[\begin{array}{cc|cc|cc} 0 & 0 & & & & \\ 0 & 1 & & & & \\ 0 & 0 & 1 & 2 & & \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & & 1 & & & \end{array}\right]$$

Figura 2.11: Ejemplo de circuito inicial en $\mathcal{C}_{4,3}$ con $m \geq 2$ inputs.

□

Escribamos, en aras de resumir este capítulo, el algoritmo entero con el que recorrer $\mathcal{C}_{n,w}$.

Algoritmo 2.7.8 (El algoritmo del índice)**.** *Recorrido de* $\mathcal{C}_{n,w}$

Algoritmo 7 El algoritmo del índice

Require: $n, w, m \in \mathbb{Z}, \eta^{n-1}(w) = 1, m \geq \sigma(w)$
  $C \leftarrow$ Circuito inicial
  $L \leftarrow [C]$
  $\sigma \leftarrow$ Mínima permutación.
  while C no sea $None$ **do** ▷ Equivalente a Fin del algoritmo.
    if Existe $\sigma' > \sigma$ **then** ▷ Podemos cambiar los inputs.
      $C \leftarrow (C \setminus \sigma) \cup \sigma$
      $\sigma \leftarrow \sigma'$
    else
      $d \leftarrow 1$
      while $d \leq n$ **do**
        $g' \leftarrow$ Incremento del tipo
        $v' \leftarrow$ Incremento del cableado
        if g no es $None$ **then**
          $C|_d \leftarrow (C|_d \setminus g) \cup g'$
          $g \leftarrow g'$
          break
        else if v no es $None$ **then**
          $g' \leftarrow$Tipo mínimo
          $C|_d \leftarrow (C|_d \setminus (v, g)) \cup (v, g')$
          $(v, g) \leftarrow (v', g')$
          break
        else if Se cumple el teorema Ampliación del suelo **then**
          $v' \leftarrow$ Vector mínimo
          $g' \leftarrow$Tipo mínimo
          $C|_d \leftarrow (C|_d \setminus (v, g)) \cup (v, g')$
          $(v, g) \leftarrow (v', g')$
          break
        end if ▷ **else** Tenemos que cambiar de clase $C|_{d+1}$
        $d \leftarrow d + 1$
      end while
      if $d = n + 1$ **then**
        $C \leftarrow None$
      else
        $C \leftarrow$Representante mínimo
      end if
    end if
    $L \leftarrow L \cup [C]$
  end while

Finalmente recalquemos que este algoritmo recorre todo el espacio por un hecho muy sencillo: $\mathcal{C}_{n,w}$ es un conjunto finito y todo $C \in \mathcal{C}_{n,w}$ está en sus respectivas clases $C|_d$, con $1 \leq d \leq n$. En concreto, podemos visualizar la existencia de una clase $\hat{C}|_{n+1}$ formada por el colapso de todos los circuitos definidos como $C' \coloneqq id(C, C), C \in \mathcal{C}_{n,w}$ y es clara la equivalencia entre nuestras pocas clases $\hat{C}|_n$ (a lo sumo 3, $\begin{bmatrix}0 & 0 & 1\end{bmatrix}$, $\begin{bmatrix}0 & 1 & 0\end{bmatrix}$ y $\begin{bmatrix}0 & 1 & 1\end{bmatrix}$) y el conjunto de clases de $\hat{C}'|_n$. Como nuestro algoritmo es determinista, el índice induce un orden sobre cada una de estas clases, y nosotros las recorremos todas ellas, en concreto, recorremos toda la clase $\hat{C}|_{n+1}$. Por ello, recorremos todo $\mathcal{C}_{n,w}$. Y decimos remarcar porque no hay duda de este hecho. Solo hemos tenido que apoyarnos en el índice y gracias a su flexibilidad y firmeza, el camino ha surgido de manera inmediata.

Capítulo 3

La endogamia de los circuitos

Una vez estudiado el espacio de circuitos y descubierto una manera de recorrerlo de manera eficiente, expondremos el problema de la endogamia y mostraremos algunas métricas que traten de medir este concepto.

3.1. El problema de la endogamia

Como se ha visto en el corolario 1.3.3, el número de circuitos generables a partir de k inputs es $(2k)^{2^n}$ (recordemos que por el teorema 1.1.23 estamos trabajando con circuitos con $m = 2k$ inputs, donde la mitad de ellos son la negación de la otra mitad). Sin embargo, el número de funciones de k bits generables es solamente de 2^{2^k}. Para k fijo, obtenemos que el conjunto $\mathcal{C}$ tiene infinitos elementos, a diferencia de las 2^{2^k} funciones booleanas obtenibles, un número finito. Por tanto, debido a que el m-polinomizador es una función total, es claro que hay alguna función que es computada por distintos circuitos.

Sin embargo, fijado n, no todas las funciones computadas por circuitos de profundidad n lo son por la misma cantidad de circuitos, e incluso puede haber funciones no computadas. Más aún, no se sabe, fijado m, qué función booleana no se computa con ningún circuito de profundidad logarítmica y anchura polinómica. Por ejemplo, si $n = 0$, solo las funciones x_i y $\neg x_i$ son computables.

Intuitivamente esto se debe a que fijados n y w, puede haber dos circuitos del nivel $l + 1$ que empleen dos veces el mismo circuito de profundidad l en su definición. A esta situación la denominamos *endogámica*, ya que se retroalimentan en exceso los niveles entre sí. Por otra parte, si dos circuitos tienen idéntica evaluación pero distinta configuración estructural, *de facto* estamos perdiendo expresividad en nuestro circuito final y también nos encontramos ante otra situación endogámica, aunque más sutil e imperceptible que la anterior.

Las diferencias entre ambas situaciones son claras: en el primer caso son resultado de la estructura que un circuito tiene, mientas que en el segundo, de la evaluación del mismo. A cada una de estas

endogamias las denominaremos *endogamia estructural* y *endogamia semántica* respectivamente. Sin embargo, en este documento solo trataremos la primera de ellas.

Para ello, presentaremos diversas métricas que traten de clasificar los circuitos según su endogamia estructural. Además, estudiaremos cómo se comportan estas sobre un mismo ejemplo para poder entender bien las diferencias y similitudes entre las mismas.

Ejemplo 3.1.1. Ejemplo con el que trabajaremos en toda esta sección, donde $m = 5, n = 4$ y $w = 4$.

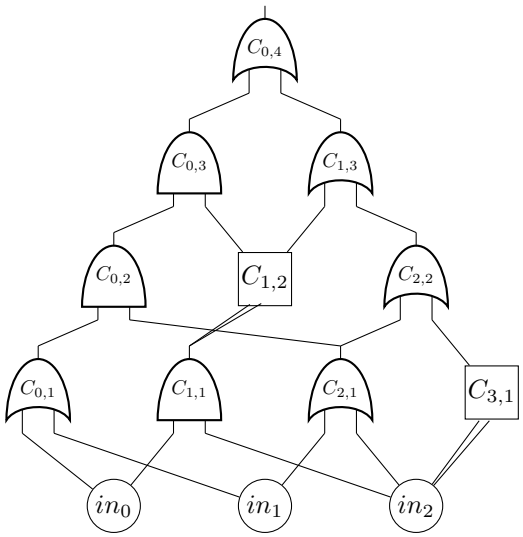


Figura 3.1: Representación física del circuito C.

$$\begin{bmatrix} (\vee,0,1) & & & \\ (\wedge,0,1) & (\vee,1,2) & & \\ (\wedge,0,2) & (id,1,1) & (\vee 2,3) & \\ (\vee,0,1) & (\wedge,0,2) & (\vee,1,2) & (id,2,2) \\ (0) & (1) & (2) & \end{bmatrix}$$

(a) Representación matricial compacta.

$$\left[\begin{array}{ccc|ccc|ccc|ccc} 0 & 1 & 0 & & & & & & & & & \\ 0 & 1 & 1 & 1 & 2 & 0 & & & & & & \\ 0 & 2 & 1 & 1 & 1 & 1 & 2 & 3 & 0 & & & \\ 0 & 1 & 0 & 0 & 2 & 1 & 1 & 2 & 0 & 2 & 2 & 1 \\ & 0 & & & 1 & & & 2 & & & & \end{array}\right]$$

(b) Representación matricial dispersa.

Figura 3.2: Representación matricial del circuito C.

Sin embargo, como se ha mencionado anteriormente, la endogamia estructural radica en las distintas conexiones entre puertas, no en el tipo de las puertas. De igual modo, qué inputs exactamente se han elegido para construir un circuito solo genera ruido al no ser la causa de esta endogamia; otra elección de inputs genera otro circuito que, en general, computa otra función, pero todos estos circuitos son idénticos estructuralmente con lo que adolecen de los mismos problemas y tienen las mismas virtudes. Notemos además que si tomamos $n = \alpha(m)$ y $w = \beta(m)$, donde α, β son dos funciones reales, es claro que el número de puertas que emplea un circuito C está en $\mathcal{O}(\alpha(m) \cdot \beta(m))$. Así, aunque α y β fueran funciones logarítmicas (con base 2), el número de circuitos generables seguirá siendo 2-exponencial. Sin embargo, el tipo de circuitos (a nivel estructural) es polinómico, y por ende, abordable computacionalmente.

Por todo ello, este análisis lo podremos realizar de manera tanto teórica como computacional trabajando en $\hat{\mathcal{C}}_{n,w}|_1$. Y así, nos basta para trabajar con la representación matricial dispersa de dicha clase.

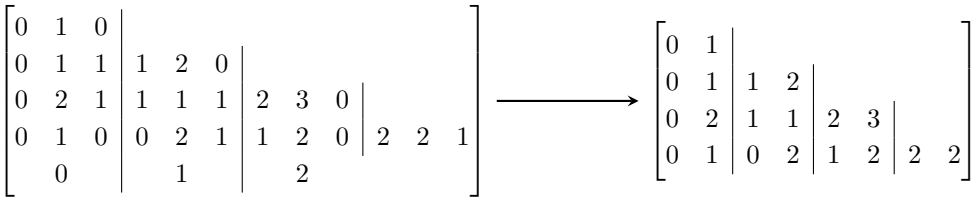


Figura 3.3: Transformación de la representación matricial dispersa del circuito C del ejemplo 3.1.1 a la transformación matricial dispersa de $\hat{C}|_1$.

3.2. Endogamia por representación

Como hemos visto en el capítulo 1, dado un circuito C en representación minimal de profundidad n cualquier representación maximal del mismo, emplea $2^n - 1$ puertas y, por ser la representación minimal del mismo emplea al menos n puertas. Eso quiere decir que cualquiera que sea el número de puertas de C, siempre estará entre n y $2^n - 1$. Además, cuanto menor sea este número, significa que más puertas se están reutilizando. En consecuencia definimos la siguiente métrica:

Definición 3.2.1. Denotamos por *endogamia por representación* de un circuito C al valor $e_r(C)$ dado por $e_r(C) = \frac{2^n - 1 - |E(C)|}{2^n - 1 - n}$.

Lema 3.2.2. *Para cualquier circuito se cumple que su endogamia por representación está entre 0 y 1.*

Demostración. Por lo expuesto anteriormente, $n \leq |E(C)| \leq 2^n - 1$. Por tanto, el resultado se sigue inmediatamente. □

Intuitivamente esta métrica mide qué porcentaje de las puertas se utilizan como input de más de un circuito. Sin embargo, hay que evitar pensar que esta métrica se deforma bien, ya que si tenemos C un circuito maximal, $e_r(C) = 0$. Sin embargo, tomando $C' \coloneqq id(C, C)$, obtenemos que $e_r(C') = \frac{2^{n+1} - 1 - 2^n}{2^{n+1} - 1 - (n+1)} = \frac{2^n - 1}{2^{n+1} - n} \approx 0{,}5$.

Ejemplo 3.2.3. Consideremos el ejemplo 3.1.1. En este caso es claro que $|E(C)| = 10$ y $n = 4$, con lo que la endogamia por representación asociada a este circuito es $e_r(C) = \frac{2^4 - 1 - 10}{2^4 - 1 - 4} = \frac{5}{11} \approx 0{,}4545$.

3.3. Endogamias vectoriales

Sin embargo, esta definición de endogamia puede resultar naïf ya que solo estudia el esqueleto de los circuitos, sin tener en cuenta las diferencias entre dos circuitos minimales con el mismo número de puertas. Por ello, plantearemos un análisis más profundo de los circuitos tratando de buscar otra métrica que refleje lo explicado anteriormente.

3.3.1. Endogamia multinivel

Sea C un circuito de profundidad n durante toda esta sección. Fijémonos en primer lugar en un nivel concreto del circuito C, por ejemplo, el nivel $l, 0 \leq l < n$. En el nivel l encontramos $|E_l(C)|$ puertas, pero no todas se emplean igual en el nivel superior. Puede que haya algunas de ellas que se empleen varias veces y otras que solamente se usen una vez. Para ello, vamos a definir nuevas funciones que nos permitan analizar con mas finura este hecho.

Definición 3.3.1. Sean C, C' circuitos tales que $C' \sqsubset C$. Denotamos por $A_C(C')$ al número de *apariciones* de C' como miembro derecho de una ecuación contenida en $E(C)$.

Destacamos de esta definición que si existe $C'' \coloneqq P(C', C'), C'' \sqsubset C$, en este caso C' aparece dos veces como miembro derecho de dicha ecuación.

Así, nos interesaría definir una endogamia por niveles de manera que se tenga en cuenta tanto el número de puertas del nivel como las apariciones de cada circuito en el mismo; tal que a mayor número de puertas, menor endogamia, y a igualdad de puertas obtener también menor endogamia cuanto más equilibradas estén las apariciones.

Lema 3.3.2. *Para cualquier $C' \sqsubset_l C$, sus apariciones están comprendidas entre 1 y $2^{n-(l+1)} + 1$.*

Demostración. Ya que C' es un circuito de profundidad l, solo puede aparecer como miembro derecho de circuitos de profundidad $l + 1$. Sin embargo, en el nivel $l + 1$ hay al menos un circuito y a lo sumo $2^{n-(l+1)}$ circuitos. Además, al menos uno de esos circuitos tiene a C' como miembro derecho, ya que si no C' no sería subcircuito y hay a lo sumo uno de ellos que puede estar definido como $C'' \coloneqq id(C', C')$. Por tanto, $1 \leq A_C(C') \leq 2^{n-(l+1)} + 1$. □

Ejemplo 3.3.3. Consideremos nuevamente el ejemplo 3.1.1 y analicemos rápidamente la matriz asociada de $\hat{C}|_1$. Además, denotaremos los circuitos como (i, j), donde i es el índice de dicho circuito y j la profundidad del mismo.

$$\left[\begin{array}{cc|cc|cc|cc} 0 & 1 & & & & & & \\ 0 & 1 & 1 & 2 & & & & \\ 0 & 2 & 1 & 1 & 2 & 3 & & \\ 0 & 1 & 0 & 2 & 1 & 2 & 2 & 2 \end{array}\right]$$

Así, observando la primera fila de la matriz es fácil ver que $A_{(0,0)}(C) = 2$, $A_{(1,0)}(C) = 2$ y $A_{(2,0)}(C) = 4$. De manera análoga, es inmediato observar en la segunda fila que $A_{(0,1)}(C) = 1$, $A_{(1,1)}(C) = 2$, $A_{(2,1)}(C) = 2$ y $A_{(3,1)}(C) = 1$, que $A_{(0,2)}(C) = 1$, $A_{(1,2)}(C) = 2$ y $A_{(2,1)}(C) = 1$ y finalmente gracias a la fila superior podemos calcular las apariciones de los circuitos de profundidad 3: $A_{(0,3)}(C) = 1$ y $A_{(1,3)}(C) = 1$. □

El ejemplo anterior ilustra claramente por qué hemos tomados subcircuitos propios en la definición de apariciones: en la representación matricial no tenemos información del circuito definido (ni nos interesa tenerla, ya que no puede ser, por definición, generado por sí mismo porque entraríamos en un bucle).

Definición 3.3.4. Sea $l \in \mathbb{N}, 0 \leq l < n$. Denotamos por *endogamia en el nivel l* de un circuito C al valor $e_l(C)$ definido con la siguiente ecuación $e_l(C) = \frac{1}{|E_l(C)|} \sum_{C' \sqsubseteq_l C} \log_{2^{n-(l+1)}+1}(A_C(C'))$.

Téngase en cuenta que la endogamia, en el sentido conceptual de la palabra, del nivel n es nula ya que siempre va a estar constituido por una única puerta lógica que no se va a utilizar más que como salida al evaluar el circuito, por tanto, no tiene sentido añadirlo a la definición al no ser de relevancia.

Lema 3.3.5. *Sea C de profundidad n. Si $l \in \mathbb{N}, 0 \leq l < n$, la endogamia en el nivel l de C está entre 0 y 1.*

Demostración. Como $1 \leq A_C(C') \leq 2^{n-(l+1)} + 1$, entonces $0 \leq \log_{2^{n-(l+1)}+1}(A_C(C')) \leq 1$ y en consecuencia, $0 \leq \frac{1}{|E_l(C)|} \cdot \log_{2^{n-(l+1)}+1}(A_C(C')) \leq \frac{1}{|E_l(C)|}$. Así, ya que hay exactamente $|E_l(C)|$ circuitos de profundidad l, obtenemos que $0 \leq \frac{1}{|E_l(C)|} \sum_{C' \sqsubseteq_l C} \log_{2^{n-(l+1)}+1}(A_C(C'_i)) \leq \sum_{C' \sqsubseteq_l C} \frac{1}{|E_l(C)|}$, es decir, $0 \leq e_l(C) \leq 1$. □

Definición 3.3.6. Dado un circuito C de profundidad n, definimos la *endogamia multinivel* como la función vectorial $\hat{e}(C) = (e_0(C), ..., e_{n-1}(C))$.

Ejemplo 3.3.7. Continuando con el ejemplo 3.1.1, calculemos $\hat{e}(C)$. Es sencillo obtener que $e_0 = \frac{1}{3}\log_{2^{4-1}+1}(2 \cdot 2 \cdot 4) \approx 0{,}4206$, $e_1 = \frac{1}{4}\log_{2^{4-2}+1}(1 \cdot 2 \cdot 2 \cdot 1) \approx 0{,}2153$, $e_2 = \frac{1}{3}\log_{2^{4-3}+1}(1 \cdot 2 \cdot 1) \approx 0{,}4206$ y $e_3 = \frac{1}{2}\log_{2^{4-4}+1}(1 \cdot 1) = 0$. Así $\hat{e}(C) = (0{,}4206, 0{,}2153, 0{,}4206, 0)$. □

Esto nos muestra que parece razonable esta definición de vector de endogamias: el nivel 4 utiliza el máximo número de circuitos que puede, 2, con lo que ese nivel es poco endogámico (0). Sin embargo, el nivel anterior utiliza 3 circuitos de los hasta 4 distintos que podría emplear, con lo que tiene sentido que tenga una endogamia "alta" (los valores de las endogamias por niveles no suelen ser valores especialmente elevados por la definición de las apariciones).

Esta definición por sí sola es agua de borrajas, es decir, no podemos evaluar un circuito como una n-tupla de endogamias por niveles de manera natural; estamos acostumbrados a trabajar en una única dimensión. Por ello, vamos a tratar de utilizar estos valores obtenidos para definir otras posibles métricas.

3.3.2. Endogamia multinivel entrelazada

Un hecho obvio es que en realidad, dado C circuito de profundidad n, el número de apariciones de un circuito de profundidad l está limitado por el número real de puertas en el nivel $l+1$, que es $|E_{l+1}(C)|$. Por ello, y razonando como en el caso anterior, podemos definir la siguiente endogamia:

Definición 3.3.8. Sea $l \in \mathbb{N}, 0 \leq l < n$. Denotamos por *endogamia entrelazada en el nivel l* de un circuito C al valor $ee_l(C)$ definido así: $ee_l(C) = \frac{1}{|E_l(C)|} \sum_{C' \sqsubseteq_l C} \log_{2|E_{l+1}(C)|}(A_C(C'))$.

El apellido *entrelazada* estriba en que la endogamia de un subcircuito de un nivel l la caracterizamos en función del número de puertas de su nivel y el siguiente, lo que conlleva que variar el

número de puertas en un nivel repercute de manera más directa en la endogamia del nivel inferior que en el caso anterior. Además, tenemos el siguiente lema:

Lema 3.3.9. *Sea C de profundidad n. Si $l \in \mathbb{N}, 0 \leq l < n$, la endogamia entrelazada en el nivel l de C está entre 0 y 1.*

Demostración. Análoga al lema anterior. □

En consecuencia y de manera análoga, podemos hablar del siguiente concepto.

Definición 3.3.10. Dado un circuito C de profundidad n, definimos la *endogamia multinivel entrelazada* como la función vectorial $\hat{ee}(C) = (ee_0(C), ..., ee_{n-1}(C))$.

Ejemplo 3.3.11. Mostremos ahora la endogamia multinivel entrelazada en el ejemplo 3.1.1. En este caso $ee_0 \approx 0{,}2187$, $ee_1 = 0{,}1250$, $ee_2 \approx 0{,}2103$ y $ee_3 = 0$. Por tanto, la endogamia obtenida es $\hat{ee}(C) = (0{,}2187, 0{,}1250, 0{,}2103, 0)$. □

Esta definición aparentemente proporciona algo más de discriminación, ya que distingue el nivel 3 del nivel 1. Con ello, las clases de circuitos obtenidos serán de un mayor espectro y se podrán analizar con mayor detalle. Sin embargo, pudiera ser que esta definición esté excesivamente ligada al circuito y dificulte la clasificación de los mismos según la función que computen, que es el objetivo último de este capítulo.

3.4. Endogamia por colapso

Sea C un circuito de profundidad n y sea $\hat{e}$ un vector de dimensión n cuyas componentes están normalizadas. Nuestro objetivo en esta y las siguientes secciones será buscar combinaciones lineales de las componentes de $\hat{e}$ de manera que la definición resultante se comporte adecuadamente según nuestros estándares.

Definición 3.4.1. Dado $v = (v_0, ..., v_{n-1}) \in \mathbb{R}^n$ decimos que v es un *vector de coeficientes* si todas sus componentes son no negativas y $\sum_{i=0}^{n-1} v_i = 1$.

Definición 3.4.2. Dado un circuito C de profundidad n, decimos que una función $f \in \mathcal{C} \to [0, 1]$ es una *endogamia por colapso ponderado* si existe un vector de coeficientes $v_C \in \mathbb{R}^n$ tal que $f(C) = v_C \cdot \hat{e}(C)$.

Intuitivamente, el apellido *por colapso* de esta definición se basa en que, al pasar un circuito de su representación maximal a su representación minimal, varias de las puertas han podido acabar resultando la misma, colapsando a un representante. Sin embargo, la ponderación de estos colapsos se deja al consumidor y eso es lo que vamos a tratar de estudiar.

Lema 3.4.3. *Sea C un circuito de profundidad n y sea v_C un vector de coeficientes asociado a C. Entonces $v_C \cdot \hat{e}(C) \in [0, 1]$.*

Demostración. Ya que $v_C = (v_{0C}, ..., v_{n-1C})$ cumple que todos sus coeficentes son menores o iguales que 1 y dado que $\hat{e}_C = (e_0(C), ..., e_{n-1}(C))$ tiene todas sus componentes menores o iguales que 1, tenemos que $v_C \cdot \hat{e} = \sum_{i=0}^{n-1} v_{iC} \cdot e_i(C) \leq \sum_{i=0}^{n-1} v_{iC} = 1$. Por otra parte, al ser tanto v_C como $\hat{e}(C)$ no negativos, se cumple que $v_C \cdot \hat{e}(C) \geq 0$. □

3.4.1. Endogamia por colapso mínimo y máximo

Una primera idea sería suponer que las endogamias por nivel son independientes y que quizás lo más sencillo sea coger el menor o el mayor valor de estos respectivamente. Así, proponemos las siguientes definiciones.

Definición 3.4.4. Definimos la *endogamia por colapso mínimo* como la función $e_{cmin} \in \mathcal{C} \to [0, 1]$, definida así: $e_{cmin}(C) = \text{mín}\{e_l(C), 0 \leq l < n\}$, donde n es la profundidad del circuito C.

Definición 3.4.5. Definimos la *endogamia por colapso máximo* como la función $e_{cmax} \in \mathcal{C} \to [0, 1]$, definida así: $e_{cmax}(C) = \text{máx}\{e_l(C), 0 \leq l < n\}$, donde n es la profundidad del circuito C.

Ambas definiciones se pueden ver como un caso particular de una endogamia por colapso ponderado tomando como ponderación al vector $v_C = (\partial_{1,k}, .., \partial_{n-1,k})$, donde k es el menor índice tal que la endogamia de C en dicho nivel es mínima (respectivamente, máxima) y $\partial_{i,k}$ es la delta de Kronecker.

Sin embargo, vamos a analizar estas definiciones brevemente para ver si son convenientes.

Lema 3.4.6. *La endogamia por colapso mínimo de un circuito de profundidad n solo toma los valores* 0 *y* $\log_{2^{n-2}+1}(2)$.

Demostración. Sea C un circuito de profundidad n. En el nivel $n-1$ hay una o dos puertas. Si hubiese dos, entonces la endogamia de dicho nivel sería 0 ya que cada circuito de profundidad $n-1$ solo podría aparecer una única vez. Si no, la endogamia de este nivel sería $\log_{2^{n-(n-1+1)}+1}(2)$ y podemos proceder de manera inductiva, suponiendo que hemos probado que $\forall l, k < l \leq n$, en el nivel l hay exactamente una puerta o bien existe un nivel intermedio en el que la endogamia es nula y continuamos razonando como en el nivel $n-1$. En un número finito de pasos hemos obtenido que o bien en algún nivel se alcanza el valor 0, o $\hat{e} = (1, \log_3 2, ..., \log_{2^{n-2}+1}(2))$. Por tanto, $e_{cmin}(C) \in \{0, \log_{2^{n-2}+1}(2)\}$. □

En definitiva, esta definición no aporta demasiado ya que discrimina de manera muy gruesa la diversidad de circuitos. Sin embargo, no podemos realizar un análisis así sobre la endogamia por colapso máximo, ya que en circuitos no extremos estos valores toman valores arbitrarios, pudiéndose alcanzar tanto la cota mínima, 0, cuando el circuito está en representación maximal hasta la cota máxima, 1, cuando el circuito tiene únicamente n puertas. Por ello, esta última puede ser considerado como una métrica razonable.

Ejemplo 3.4.7. Calculemos rápidamente la endogamia por colapso mínimo y por colapso máximo en el ejemplo 3.1.1. En este caso, $e_{cmin}(C) = 0$ y $e_{cmax}(C) = 0{,}4206$. □

Aún así, estas dos definiciones generan dudas en cuanto a si son capaces de recoger toda la expresividad que la definición de circuito tiene. Por ello, vamos a proponer nuevas definiciones de endogamia.

3.4.2. Endogamia por colapso directo

Una idea bastante razonable es suponer que no todos los tipos de circuitos influyen de la misma manera. Por ejemplo, si un circuito de profundidad 1 es utilizado por un gran número de circuitos de profundidad 2 es claro que el cambio de valor en su evaluación puede afectar mucho más en el resultado final. Este puede no se debe tomar como un axioma ni como una variable probabilística, sino como la intuición que guiará esta definición. Además, cuanto más aumentemos la profundidad de los circuitos, un cambio en la evaluación de dicho circuito se propagará a priori a menos circuitos. Por todo ello:

Definición 3.4.8. Definimos la *endogamia por colapso directo* como la función $e_{cd} \in \mathcal{C} \to [0,1]$, definida así: $e_{cd}(C) = v_d \cdot \hat{e}(C)$, donde $v_d = \frac{1}{1-2^{-n}}(2^{-1}, 2^{-2}, ..., 2^{-n})$ y n es la profundidad del circuito C.

Es claro que v_d es un vector de coeficientes ya que la suma de sus componentes es 1. Además, refleja fielmente lo explicado anteriormente.

3.4.3. Endogamia por colapso inverso

Sin embargo, podemos realizar un razonamiento inverso al anterior: un cambio en la evaluación de un circuito de profundidades muy elevadas es más probable que se refleje en un cambio real en la evaluación del circuito de manera global. Por ello, quizás nos interese ponderar de manera inversa a lo anterior, focalizándonos en las endogamias superiores y no las inferiores.

Definición 3.4.9. Definimos la *endogamia por colapso inverso* como la función $e_{ci} \in \mathcal{C} \to [0,1]$, definida así: $e_{cd}(C) = v_i \cdot \hat{e}(C)$, donde $v_i = \frac{1}{1-2^{-n}}(2^{-n}, 2^{-(n-1)}, ..., 2^{-1})$ y n es la profundidad del circuito C.

Es además inmediato por lo dicho anteriormente que v_i es un vector de coeficientes. Nótese que en ambos casos se han tomado coeficientes de la forma 2^{-k} solamente por el hecho de que su representación maximal es un árbol equilibrado donde el número de puertas es siempre potencia de dos; podríamos haber elegido otro vector de coeficientes y haber razonado de manera análoga con estas dos definiciones.

Ejemplo 3.4.10. La endogamia por colapso directo asociado al ejemplo 3.1.1 es $e_{cd}(C) \approx 0{,}3378$ y la endogamia por colapso inverso resulta $e_{ci}(C) \approx 0{,}1689$. Intuitivamente esto quiere decir que si estimamos que la causa de la endogamia está en los niveles superiores, este circuito no lo es, ya

que el primer y el segundo nivel se expanden casi en su totalidad. Sin embargo, si nos fijamos en la endogamia directa, tiene más sentido un "alto" valor ya que hay a la postre pocos nodos. □

3.4.4. Endogamia por colapso entrelazada

Otra idea que puede ser interesante es tratar de medir la propagación de manera directa e inversa, pero teniendo en cuenta la endogamia entrelazada. Así, podemos realizar un análogo al caso anterior con estas definiciones.

Definición 3.4.11. Definimos la *endogamia por colapso directo entrelazada* como la función $e_{cde} : \mathcal{C} \to [0,1]$ definida como: $e_{cde}(C) = v_d \cdot \hat{ee}(C)$, donde v_d es el vector de coeficientes de la definición 3.4.8.

Definición 3.4.12. Definimos la *endogamia por colapso inverso entrelazada* como la función $e_{cie} : \mathcal{C} \to [0,1]$ definida como: $e_{cie}(C) = v_i \cdot \hat{ee}(C)$, donde v_i es el vector de coeficientes de la definición 3.4.9.

Finalmente, trataremos de refinar la idea directa teniendo en cuenta, para cada nivel l, a cuántos circuitos del nivel $l+1$ podría afectar en realidad un cambio en la evaluación del circuito.

Definición 3.4.13. Definimos la *endogamia por colapso bi-entrelazado* como la función $e_{cbe} : \mathcal{C} \to [0,1]$ definida como: $e_{cbe}(C) = v_b(C) \cdot \hat{ee}(C)$, donde $v_b(C) = \frac{1}{\sum_{l=1}^{n} |E_{l+1}(C)|}(|E_1(C)|, ..., |E_n(C)|)$

Ejemplo 3.4.14. Calculemos pues las endogamias por colapso definidas utilizando las endogamias entrelazadas con el ya recurrente ejemplo 3.1.1. Los valores obtenidos son: $e_{cde}(C) \approx 0{,}1780$, $e_{cie}(C) \approx 0{,}0873$ y $e_{cbe}(C) \approx 0{,}1670$.

Por una parte, obtener que $e_{cde}(C) > e_{cie}(C)$ es lo esperado al poder realizar la misma argumentación que en el ejemplo anterior. Por otra parte, por la naturaleza del vector v_d y la del $v_b(C)$, los cuales ponderan el número de circuitos a las que afecta un cambio en un nivel, tiene sentido que se dé que $e_{cde}(C) \approx e_{cbe}(C)$. □

De este ejemplo surge una pregunta interesante: ¿introducir el concepto de entrelazamiento genera ruido generando más clases de circuitos o proporciona distinción sin perder la potencia de la definición de colapso directo?

3.5. Umbral de colapso

Una pregunta natural de todas estas definiciones es cómo varía la endogamia por colapso de los circuitos en función de los vectores de coeficientes, o lo que es lo mismo, si existe alguna manera de entender cómo poder discriminar utilizando estas endogamias y cómo afectan estas modificaciones.

Definición 3.5.1. Se denomina *permutación* a cualquier función biyectiva $\sigma \in X \to X$.

Nosotros tomaremos usualmente como $X = \{1, ..., n\}$. Además, dado un vector $v = (v_1, ..., v_n)$ en $\mathbb{R}^n$ y una permutación σ, abusaremos de la notación y hablaremos del vector $\sigma(v) = (v_{\sigma(1)}, ..., v_{\sigma(n)})$.

Definición 3.5.2. Sea $\lambda = (\lambda_1, ..., \lambda_n)$. Se denomina *$\lambda$-umbral mínimo* a la función $u_\lambda \in \mathcal{C} \to [0, 1]$ que minimiza $\sigma(\lambda) \cdot \hat{e}(C)$, con σ permutación.

Definición 3.5.3. Sea $\lambda = (\lambda_1, ..., \lambda_n)$. Se denomina *$\lambda$-umbral máximo* a la función $U_\lambda \in \mathcal{C} \to [0, 1]$ que maximiza $\sigma(\lambda) \cdot \hat{e}(C)$, con σ permutación.

Teorema 3.5.4 (Desigualdad de reordenamiento)**.** *Sean $x_1 \leq \cdots \leq x_n$ y $y_1 \leq \cdots \leq y_n$ dos colecciones de n valores ordenados. Entonces $x_1 y_n + \cdots + x_n y_1 \leq x_1 y_{\sigma(1)} + \cdots + x_n y_{\sigma(n)} \leq x_1 y_1 + \cdots + x_n y_n$, donde σ es una permutación cualquiera. [4]*

Corolario 3.5.5. *Dado λ vector de coeficientes de tamaño n, las funciones u_λ y U_λ se pueden obtener para cada circuito en tiempo en $\mathcal{O}(n \log(n))$.*

Demostración. De la desigualdad de reordenamiento obtenemos que nos basta con ordenar el vector λ y el vector $\hat{e}(C)$ y multiplicar dichos vectores ordenados adecuadamente. Y eso es claro que se puede obtener en tiempo $\mathcal{O}(n \log(n))$. □

Definición 3.5.6. Denominamos *λ-umbral de colapso* de un circuito al intervalo $[u_\lambda(C), U_\lambda(C)]$.

De todo lo explicado anteriormente nos damos cuenta de que las endogamias por colapso mínimo y máximo no son sino permutaciones del vector $v' = (1, 0, .., 0)$, al igual que lo son las endogamias por colapso directo e indirecto con $\lambda = v_d$. De hecho, $e_{cmin} = u_{v'}$ y $e_{cmax} = U_{v'}$.

Podemos dar la vuelta a este concepto y ver, fijado un vector de coeficientes, cómo varía la endogamia en función de la endogamia de cada nivel. Por ejemplo, en la endogamia por colapso directo, no es lo mismo que los niveles superiores tengan valores altos de endogamia a que sean los inferiores los de alta endogamia; en el primer caso obtendríamos un valor de endogamia por colapso directo mucho menor que en el segundo.

Capítulo 4

Gramáticas y circuitos

En este capítulo retomaremos el estudio teórico del espacio de circuitos mediante las funciones booleanas que computan, es decir, analizando $ev_{\#}(C)$, en aras de buscar relaciones entre los circuitos y las funciones que computan.

Recordemos que uno de los conceptos que queríamos estudiar es la relación entre la repetitividad en el patrón de una función y los circuitos que computan estas funciones. Sin embargo, esta idea puede ser observada de muchas maneras. Por ejemplo, la función 00001111000011110000111100001111 se puede ver como repetir cuatro veces el patrón 00001111 o repetir 01111000 tres veces con prefijo 000 y sufijo 01111. Sin embargo, nuestra visión de la repetitividad estará basada en la bisección reiterada del patrón, buscando una única manera de expresar este fenómeno que simplifique, en general, la expresión del mismo. Para ello, nos valdremos del concepto de gramática libre de contexto, un manera de conseguir funciones booleanas de naturaleza muy distinta a la trabajada hasta ahora con los circuitos, pero que sin embargo no son del todo ajenas la una de la otra.

4.1. Gramáticas libres de contexto

Recordemos en primer lugar el concepto de gramática libre de contexto.

Definición 4.1.1. Denominamos *grámatica libre de contexto* a la tupla $G = (\mathcal{V}, \Sigma, \mathcal{P}, S)$ donde:

- $\mathcal{V}$ es el conjunto de variables,
- Σ es el conjunto de terminales,
- $\mathcal{P}$ es el conjunto de producciones,
- S es la variable inicial.

La principal aplicación de estas gramáticas viene recogida en el siguiente resultado:

Teorema 4.1.2. *Todo elemento f de ${\mathbb{F}_2}^{2^n}$ se puede obtener mediante una gramática libre de contexto con $\Sigma = \{0,1\}$ y un conjunto de variables y producciones irredundantes, es decir, conjuntos cuyos elementos son distintos dos a dos.*

Demostración. La demostración de este teorema será constructiva e inductiva. En primer lugar, parece razonable tomar dicho conjunto de terminales ya que los elementos de $\mathbb{F}_2$ son $\{0,1\}$. Además, denotaremos $int(f), f \in {\mathbb{F}_2}^{2^n}$ al número $\tilde{f}(2)$, donde $\tilde{f}$ es el polinomio en $Z[x]$ cuyos coeficientes son $f = (f_0, \ldots, f_{2^n})$ (de manera que los elementos 0 y 1 de $\mathbb{F}_2$ se correspondan con el 0 y el 1 de $\mathbb{Z}$).

Sea $n = 0$. Entonces $f = 0$ o $f = 1$ y definamos $P_{0,int(f)} = V_{0,int(f)} \to f$. Así, tomemos $\mathcal{V} = \{V_{0,int(f)}\}, \mathcal{P} = \{P_{0,int(f)}\}$ y $S = P_{0,int(f)}$. Es inmediato observar que esta gramática genera f y que satsiface las hipótesis del teorema.

Así, razonemos por inducción, supuesto que todo elemento de $\mathbb{F}_2^{2^n}$ es generable por una gramática como en el enunciado. Dado $f \in \mathbb{F}_2^{2^{n+1}}$, f es una 2^{n+1}-tupla de elementos de $\mathbb{F}_2$, con lo que denotaremos por $f_1 = (f_{2^{n+1}-1}, ..., f_{2^n})$ y $f_2 = (f_{2^n-1}, ..., f_0)$, de manera que con $f = f_1 \cdot f_2$ denotaremos la concatenación de tuplas, no el producto. Por hipótesis de inducción, existen dos gramáticas $G_1 = (\mathcal{V}_1, \Sigma, \mathcal{P}_1, S_1)$ y $G_2 = (\mathcal{V}_2, \Sigma, \mathcal{P}_2, S_2)$ de manera que generan f_1 y f_2 respectivamente. Por tanto, definamos $P_{n+1,int(f)} = V_{n+1,int(f)} \to V_{n,int(f_1)} \cdot V_{n,int(f_2)}$ y observemos que denotando $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \{P_{n+1,int(f)}\}$, $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \{V_{n+1,int(f)}\}$ y $S = P_{n+1,int(f)}$, obtenemos que $G = (\mathcal{V}, \Sigma, \mathcal{P}, S)$ es una gramática libre de contexto que genera f y se cumple que $\mathcal{V}$ y $\mathcal{P}$ son irredundantes gracias a la notación aplicada en variables y producciones (junto a la definición de conjunto, que no permite multiplicidad). □

Ejemplo 4.1.3. Sea $n = 3$ y $f = 00011010$. Así, el conjunto de producciones que generan dicha función es:

		$V_{3,26} \to V_{2,1}V_{2,10}$	*00011010*		
$V_{2,1} \to V_{1,0}V_{1,1}$	*0001*	$V_{2,10} \to V_{1,2}V_{1,2}$	*1010*		
$V_{1,0} \to V_{0,0}V_{0,0}$	*00*	$V_{1,1} \to V_{0,0}V_{0,1}$	*01*	$V_{1,2} \to V_{0,1}V_{0,0}$	*10*
$V_{0,0} \to 0$	*0*	$V_{0,1} \to 1$	*1*		

donde en cursiva hemos indicado para cada producción $P_{n,k}$ qué función de tamaño 2^n genera.

Para finalizar la sección, ya que hay una interdependencia entre variables y producciones, remarcaremos que realizaremos un abuso del lenguaje, ya que hablaremos indistintamente de ambos conceptos en aras de hacer más liviana la siguiente sección.

4.2. Operadores no simétricos y el teorema de equivalencia

Nuestro objetivo, como buenos matemáticos, consistirá en reducir el estudio de las gramáticas al caso anterior de los circuitos, para aprovechar sus nociones y construcciones. Para ello, observemos

que las gramáticas expuestas en el teorema 4.1.2 tiene una apariencia de circuitos. Sin embargo, no es tan sencillo comno se querría.

Definición 4.2.1. Diremos que un operador $P \in \mathcal{C} \times \mathcal{C} \to \mathcal{C}'$ es *no simétrico* si no es simétrico para todo $C_1, C_2 \in \mathcal{C}, C_1 \neq C_2$.

Aunque esta definición parece innecesaria, nos sirve para poner el foco en lo siguiente: $\mathcal{V}$ se puede construir de manera idéntica a $\mathcal{C}$ con un poco más de esfuerzo.

Definición 4.2.2. Denotaremos por $\mathcal{V}_n$ al conjunto de variables $V_{n,k}$ empleables por la gramática definida en el teorema 4.1.2 para generar el elemento $f \in {\mathbb{F}_2}^{2^n}$, donde $k = int(f)$.

Lema 4.2.3. *El conjunto $\mathcal{V}_n$ tiene 2^{2^n} elementos.*

Demostración. Es inmediato debido a que $\mathcal{V}_n$ está en biyección con ${\mathbb{F}_2}^{2^n}$. □

Definición 4.2.4. Denotaremos por $ev_{\#_0} : \mathcal{V}_0 \to \mathbb{F}_2$ al operador evaluación: $ev_{\#_0}(V_{0,k}) = k$.

Definición 4.2.5. Denotaremos por $ev_{\#_n} : \mathcal{V}_n \to {\mathbb{F}_2}^{2^n}$ al operador evaluación $ev_{\#_n}(V_{n,k}) = x_n \cdot ev_{\#_{n-1}}(V_{n-1,k_1}) + ev_{\#_{n-1}}(V_{n-1,k_2})$. Además denotaremos por $\mathcal{E}v_\#$ a $\mathcal{E}v_\# = \{ev_{\#_n}, n \in \mathbb{N}\}$.

Así, observaremos que podemos definir la siguiente puerta lógica:

Definición 4.2.6. Denotaremos por $\cdot \in \mathcal{V}_n \times \mathcal{V}_n \to \mathcal{V}_{n+1}$ a la puerta lógica *concatenación*, de manera que el equivalente de dicha puerta sea $eq(\cdot) \in {\mathbb{F}_2}^{2^n} \times {\mathbb{F}_2}^{2^n} \to {\mathbb{F}_2}^{2^{n+1}}$ definido así: $eq(\cdot)(f,g) = eq(\cdot)((f_{2^n-1}, \dots, f_0), (g_{2^n-1}, \dots, g_0)) = (f_{2^n-1}, \dots, f_0, g_{2^n-1}, \dots, g_0)$, de manera que se cumple que $(ev_{\#_{n+1}} \circ \cdot)\ (V, V') = eq(\cdot)(ev_{\#_n}(V), ev_{\#_n}(V'))$.

Como se puede ver, gracias a la flexibilidad de la definición asociada al operador evaluación, podemos de una manera muy cómoda hablar de la puerta lógica concatenación y en consecuencia ver el espacio de variables $\mathcal{V}$ como un espacio de circuitos construidos con el operador $\cdot$. Sin embargo, esta construcción difiere de la usual tanto en el número de operadores como en el tipo de puerta, al ser claro que la concatenación no es simétrica. Por ello, un objetivo razonable sería buscar una manera de transformar $\mathcal{V}$ en $\mathcal{C}$ para poder utilizar todas las herramientas mencionadas anteriormente. En concreto, teniendo en mente el objetivo del experimento, de lograr esta transformación podremos aprovecharnos del algoritmo 2.7.8 para recorrer de manera exhaustiva $\mathcal{V}$.

Teorema 4.2.7 (Teorema de equivalencia)**.** *Todo elemento de $\mathcal{V}$ está en correspondencia con uno de $\mathcal{C}$ en representación minimal y bien conectado cuando $m = 2$. Además, se puede lograr esta biyección compatible con el índice en $\mathcal{C}$.*

Demostración. En primer lugar, por el lema 4.2.3 y el corolario 1.3.3, es claro que para todo $n \in \mathbb{N}$ se da que $|\mathcal{V}_n| = |\mathcal{C}_n| = 2^{2^n} < \infty$, con lo que de una manera cristalina observamos que dicha biyección existe y en consecuencia, el grueso del teorema queda ya probado. Sin embargo, mostraremos cómo, de todas las biyecciones existentes, elegir una biyección ϕ compatible con el índice.

Si $n = 0$, podemos tomar $\phi(in_0) = V_{0,0}$ y $\phi(in_1) = V_{0,1}$. Así, $\mathcal{V}_0 \approx \mathcal{C}_0$ y podemos hablar de la función índice f_0 sobre $\mathcal{V}_0$. Apliquemos ahora inducción sobre n, de manera que $\mathcal{V}_n \approx \mathcal{C}_n$ y podamos

hablar de f_n sobre $\mathcal{V}_n$. En ese caso, dado $C' \coloneqq P(C_1, C_2)$ de profundidad $n+1$, pueden darse tres casos:

- $\underline{P = id}$. En este caso $C_1 = C_2$ y en consecuencia, sus índices coinciden. Así, denotando $k = 2^n f_n(C_1) + f_n(C_2)$, podemos definir $\phi(C') = V_{n+1,k}$.
- $\underline{P = \vee}$. En este caso $f_n(C_1) < f_n(C_2)$ por la buena conexión de C'. Así, denotando arbitrariamente $k = 2^n f_n(C_1) + f_n(C_2)$, podemos definir $\phi(C') = V_{n+1,k}$.
- $\underline{P = \wedge}$. En este caso también se da que $f_n(C_1) < f_n(C_2)$. Así, denotaremos $k = 2^n f_n(C_2) + f_n(C_1)$, y definiremos $\phi(C') = V_{n+1,k}$.

Esta aplicación es claramente biyectiva y respeta el índice en el sentido siguiente: dados $V_1, V_2, V_3 \in \mathcal{V}_n$ tales que $V_1 \leq V_2 \leq V_3, V_1 \neq V_3$, se da que $V_1V_2 \leq V_2V_1 < V_1V_3 < V_3V_1$ (donde la primera desigualdad es una igualdad si $V_1 = V_2$). Esto es importante porque esta desigualdad hace corresponder a $\mathcal{V}$ una función índice que formalmente se define de manera análoga a la de $\mathcal{C}$ y, lo que es más importante, nos permite recorrer $\mathcal{V}$ con el algoritmo 2.7.8 de una manera natural: solo hay que interpretar el circuito obtenido como una variable. □

El teorema 4.2.7 se puede enunciar así: en un espacio $\mathcal{C}$ con un orden total todo operador no simétrico $P : \mathcal{C} \times \mathcal{C} \to \mathcal{C}'$ se corresponde con dos operadores simétricos $P_1, P_2 : \mathcal{C} \times \mathcal{C} \to \mathcal{C}'$ tales que para todo $C \in \mathcal{C}, P_1(C, C) = P_2(C, C)$. Sin embargo, hemos decidido exponerlo con un enunciado más simplificado para ver su aplicación inmediata, aunque ello implique que se difumine la idea de equivalencia implícita en su apellido. Además, gracias a la biyección construida en dicho teorema, los conceptos de endogamia expuestos en el capítulo 3 no solo se pueden aplicar, sino que tiene sentido estudiarlos, ya que estructuralmente es lo mismo un circuito que una producción.

Por todo ello surgen varias preguntas que no pudimos realizar en dicho capítulo al no haber hablado de estos conceptos: ¿hay alguna relación entre la endogamia de un circuito que computa una función y la producción que genera dicha función? ¿la gramática de una función determina el mínimo número de puertas de un circuito que compute la misma?

Capítulo 5

Resultados y conclusiones

Finalizaremos el trabajo recapitulando el experimento realizado, exponiendo los resultados obtenidos y extrayendo alguna conclusión de los mismos.

Como ya se mencionó en la introducción, el experimento ha consistido en el recorrido exhaustivo del espacio de circuitos, donde para cada circuito generado se ha obtenido la función que este computa y se actualiza una tabla dispersa en la que anotamos para cada función el menor circuito que la computa. Además, se ha recorrido de igual manera el espacio de las gramáticas asociadas a funciones de 5 bits, para finalmente tratar todos los datos obtenidos de manera estadística estudiando distintas correlaciones entre variables aleatorias basadas en los conceptos de endogamia y el número de puertas del circuito / reglas de la gramática.

5.1. Implementación del algoritmo del índice

Para poder realizar nuestro experimento, hemos realizado una implementación del algoritmo 2.7.8 en C++ [5]. Por claridad del código, se decidió separar en dos secciones de código diferenciadas la generación de circuitos de la evaluación de los mismos. Así, se representó cada circuito en su representación matricial compacta y el circuito en su representación física, como un árbol de punteros compartidos [6].

La necesidad de contar con ambas definiciones tiene la siguiente motivación: el algoritmo del índice trabaja con matrices, mas la evaluación de un circuito nace con esta expresión. Del paso de una expresión a la otra surge la necesidad de trabajar con la expresión matricial compacta, una expresión intermedia entre ambas representaciones; y para paliar los problemas que surgen de manera inherente a la implementación de un árbol de punteros, se ha utilizado la librería [7] para detectar fugas de memoria.

Sin embargo, como se vio en el corolario 1.3.3, el número de circuitos de m bits crece de manera doblemente exponencial con la profundidad; por lo que aun restringiendo w el tiempo de ejecución

está bastante limitado por el tamaño del espacio. Por ello, se ha decidido aplicar técnicas de programación concurrente que nos permitan ejecutar distintas secciones en paralelo. No obstante, la generación y destrucción de hilos tiene un coste, por lo que, para poder amortizar su aplicación, el tiempo del proceso paralelizado debe ser muy superior al tiempo de coste del hilo.

Por ello, se ha optado por fijar un tamaño de tarea de 1000 circuitos, de manera que el coste de un hilo fuese el coste de evaluar los 1000 circuitos. Además, para reducir aún más el coste de generación y destrucción de los hilos, se ha decidido utilizar el patrón *threadpool* [8] con el objetivo de reutilizar dichos hilos lo máximo posible. Este patrón no ha sido implementado, sino que hemos tomado una librería preexistente [9], la cual ha funcionado perfectamente.

Por otra parte, para poder analizar no solo los circuitos que aparecen y las funciones que son computadas, sino también cuál es la distribución de funciones computadas y su frecuencia, hemos necesitado de un diccionario que admita números con precisión infinita. Por el lenguaje seleccionado, que permite gran velocidad de ejecución a cambio de menor procesamiento de información en las librerías estándar, hemos requerido de una librería construida *ad hoc* para ello [10].

Sin embargo, estas decisiones no han sido las únicas tomadas en dicha implementación. Debido a la previsible necesidad de portabilidad del código, ha sido necesario crear un Makefile que lo posibilite. Así, por comodidad y sencillez en su desarrollo, se ha optado por aprender el uso de la herramienta Automake [11]. De esta manera, hemos desarrollado una implementación eficiente del algoritmo del índice, que explota toda la capacidad de cómputo de la máquina utilizando todos sus procesadores sin carrera de datos y con generación de back-up y posibilidad de restaurarlo.

Además, se ha diseñado una modificación rápida de dicho software para poder generar las gramáticas mencionadas en el capítulo 4, aprovechando la gran mayoría de lo implementado gracias al teorema 4.2.7 Este software se ha corrido en una máquina virtual del Instituto de Física Teórica (IFT), centro asociado a la UAM, por la exigencia de tiempo de cómputo que ha requerido el experimento.

Finalmente, se ha realizado el tratamiento de datos en Python aprovechando la versatilidad de la librería pandas [12] que permite la gestión cómoda de grandes cantidades de datos y el tratamiento estadístico de los mismos; destacando entre todas sus utilidades el cálculo del coeficiente de correlación de Pearson.

5.2. Información analizada

Antes de intentar responder a las preguntas que hemos ido dejando abiertas a lo largo de los capítulos 3 y 4, destaquemos una vez más aquellas palabras que acompañaban la introducción y motivación: todos los resultados obtenidos no son asintóticos para distintos valores de m, n o w, son meros frutos del estudio de un escaso conjunto de circuitos para pequeños valores de n, w con m fijo y exactamente 5. Así, todos los resultados que a continuación se presenten no son más que fruto del intento de obtener evidencias empíricas de algún tipo de relación entre los datos obtenidos, con el objetivo de abrir una nueva vía de estudio sobre si dichas evidencias constituyen parte de una realidad mayor.

Ejemplo 5.2.1. Diremos que un circuito C sobre m inputs tiene forma *romboidal* si en el nivel

$l > 0$ se cumple que $|E_l(C)| = \delta_m(l)$ y tiene profundidad $n = l + \lceil \log_2(\delta_m(l)) \rceil$. Este tipo de circuitos, por construcción, tienen los primeros l niveles con altos niveles de endogamia, al utilizarse todas las puertas la mayor cantidad de veces posible y los últimos con escasa o nula, ya que por la profundidad elegida se comportan casi como un árbol equilibrado como es fácil de comprobar.

Ejemplo 5.2.2. Diremos que un circuito C sobre m inputs tiene forma *4-romboidal* si en el nivel 1 se cumple que $|E_l(C)| = \delta_m(1)$ y en los siguientes niveles el circuito se puede ver como 4 circuitos romboidales de misma anchura sobre $\lfloor \frac{1}{4}|E_1(C)| \rfloor$ inputs.

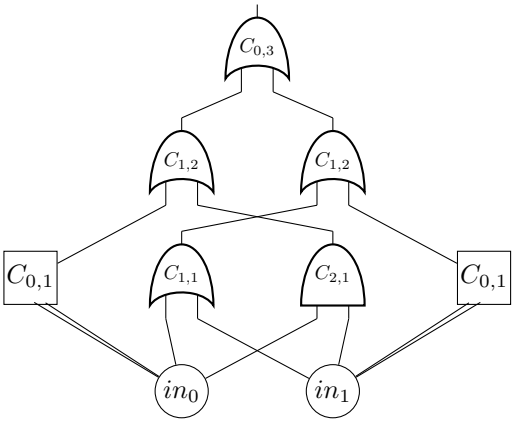


Figura 5.1: Ejemplo de circuito romboidal con $m = 2, l = 1$.

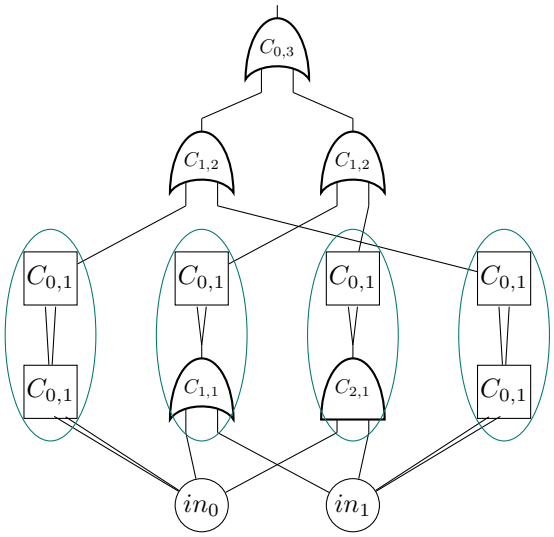


Figura 5.2: Ejemplo de circuito 4-romboidal. En verde los 4 circuitos romboidales (sobre $m = 1$).

Así, con el objetivo de dar al lector una visión completa, vamos a enumerar los tipos de circuitos generados y analizados a lo largo de toda nuestra experimentación.

- Todos los circuitos de cualquier forma posible cuyo par profundidad-anchura (n, w) está en el conjunto $\mathcal{W} = \{(0,1),(1,1),(2,1),(3,1),(2,2),(4,1),(3,2),(5,1),(4,2),(3,3),(6,1),(5,2),(4,3),(3,4),(7,1),(6,2)\}$, en total, 578269610 circuitos. De entre estos circuitos, hemos trabajado con los de menor número de puertas para misma función booleana computada. A estos los denominaremos circuitos mínimos.
- Algunos de los circuitos de profundidad 5 y anchura 3 (en concreto, $1{,}6 \times 10^{10}$ circuitos).
- Todas las gramáticas de profundidad 5, 2 bits de input (x_0 y $\neg x_0$) y anchura entre 1 y 3.
- Algunas de las gramáticas de profundidad 5, 2 bits de input y anchura en el conjunto $\{4, 8\}$.
- 10^4 circuitos generados aleatoriamente con forma romboidal.
- 10^4 circuitos generados aleatoriamente con forma 4-romboidal.

Todas las imágenes que siguen a esta sección se exponen individualmente en un anexo para poder ser analizadas con detenimiento. Puede navegar cómodamente por el documento gracias a los hipervínculos para visualizar en detalle las imágenes.

5.3. Funciones alcanzadas

En primer lugar, hemos decidido realizar una criba entre los datos obtenidos. Para ello, dentro de las gramáticas y circuitos aleatorios estudiados, se ha decidido solo analizar el subconjunto de los mismos para los cuales contábamos con un circuito mínimo que computaba la misma función booleana que la gramática o el circuito. Además, llamaremos *id decimal* de una función booleana f al número $int(f) = \tilde{f}(2)$ cuya definición precisa apareció en la demostración del teorema 4.1.2.

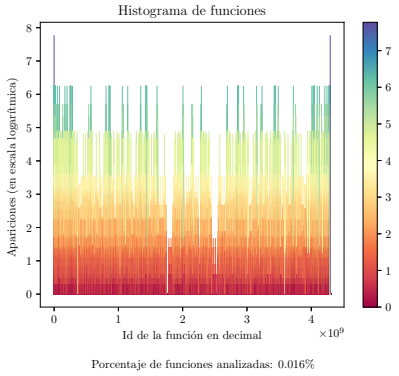


Figura 5.3: Histograma de funciones computadas (casos completos). Ampliar

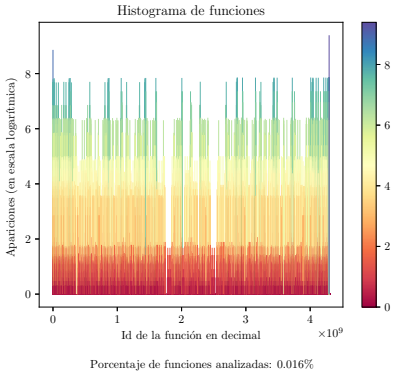


Figura 5.4: Histograma de funciones computadas (casos completos e incompletos). Ampliar

Estos histogramas representan el número de funciones computadas por circuitos obtenidos en el recorrido exhaustivo y sistemático de $\mathcal{C}$. En concreto, denominamos caso completo a todo par (n, w) de manera que todo circuito de $\mathcal{C}_{n,w}$ ha sido generado, y llamamos caso incompleto al par (n, w) tal que los circuitos de $\mathcal{C}_{n,w}$ han sido generados de manera sistemática pero dicho espacio no se ha explorado en su totalidad. Es decir, un caso completo es un par perteneciente al conjunto $\mathcal{W}$ definido en la sección anterior y el único caso incompleto tratado es el par $(5, 3)$.

Como se puede ver, no todas las funciones son alcanzadas, ni todas ellas en la misma proporción. De hecho, destacan de manera notable las funciones $\top$ y $\bot$, las cuales se han localizado 59354805 veces, lo que les da un valor de 7,77 en escala logarítmica. Además, el caso incompleto (figura 5.4) no solo no preserva la simetría de la gráfica 5.3, ya que nunca hemos garantizado dicha propiedad en nuestro algoritmo, sino que el número de funciones obtenidas es idéntico. Esto no es un error de aproximación, se han visitado *exactamente* las mismas funciones a pesar de haber visitado más de 27 veces más circuitos en el conjunto de casos completos que en el caso incompleto; el caso $(5, 3)$ no aportó ninguna sola función booleana nueva, algo insólito ya que intuitivamente esperábamos

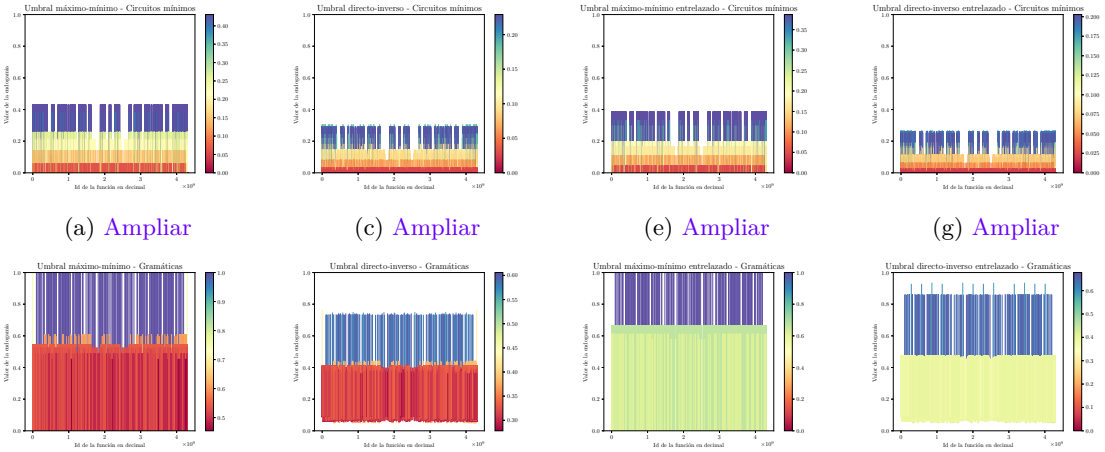
que las posibles nuevas funciones computadas en este caso deberían estar dispersas por $\mathcal{C}_{5,3}$. Este hecho resalta una vez más la existencia de una endogamia inherente en los circuitos, y por tanto, la justificación de este experimento.

	Circuitos romboidales	Circuitos 4-romboidales	Gramáticas
Recorridos	10000	10000	$2{,}63 \times 10^8$
Con circuito mínimo asociado	8281	7690	58806
Eficiencia del dataset	82,810 %	76,900 %	0,022 %
Sin repetición de función	1491	1794	58806
Eficiencia real del dataset	14,910 %	17,940 %	0,022 %

Como se puede ver en la tabla, generar gramáticas en general es más costoso computacionalmente y proporciona pocas gramáticas con circuito mínimo computado. Sin embargo, generar circuitos aleatorios con las formas elegidas es engañosamente eficiente, al haber repeticiones indeseadas (pero por otra parte esperadas). Notemos finalmente que solo se han seleccionado los datos sin repetición allá dónde ha sido necesario, y en dicho caso se ha elegido como representante aquel circuito con menor endogamia de entre todos los que entran en conflicto.

5.4. Análisis del λ-umbral

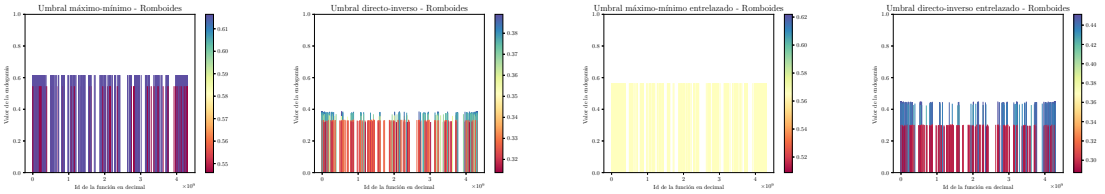
Una de las métricas planteada en el capítulo 3 fue el concepto de λ-umbral. Así, hemos estudiado los distintos umbrales para cada tipo de endogamia y cada dataset, otorgándole un color característico en función de su diferencia, de manera que se vea no solo la altura sino la diferente densidad de cada tipo de altura.



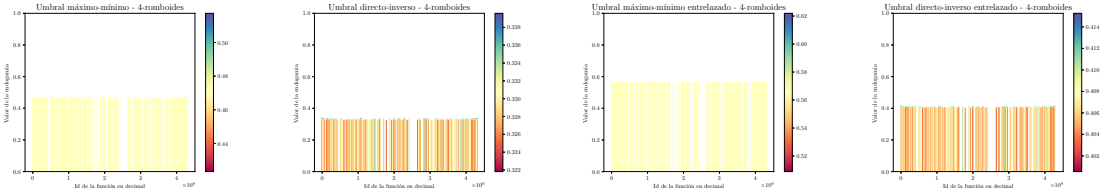
(a) Ampliar (c) Ampliar (e) Ampliar (g) Ampliar

(b) Ampliar (d) Ampliar (f) Ampliar (h) Ampliar

Figura 5.5: Distintas gráficas de λ-umbral para circuitos mínimos y gramáticas.



(a) Ampliar (c) Ampliar (e) Ampliar (g) Ampliar



(b) Ampliar (d) Ampliar (f) Ampliar (h) Ampliar

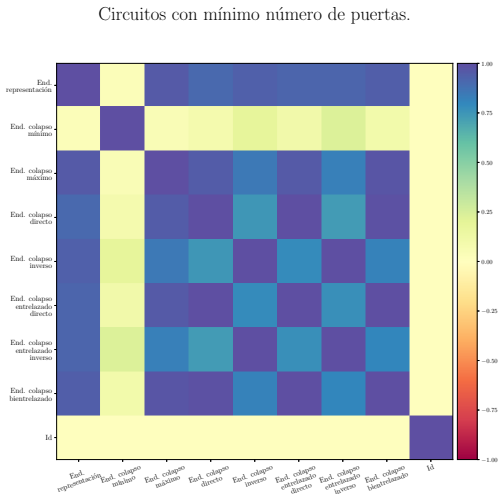
Figura 5.6: Distintas gráficas de λ-umbral para circuitos romboidales y 4-romboidales.

A la vista de estos resultados es claro que esta métrica permite de manera visual discriminar si un conjunto de circuitos arbitrario es mínimo o tiene alguna estructura particular. Por ejemplo, los circuitos mínimos tienen un máximo umbral directo-inverso máximo entorno a 0,2, y los distintos umbrales están estratificados en unas cuatro categorías. Sin embargo, el caso de las gramáticas, el umbral directo-inverso estratifica en dos tipos de circuitos, los de umbral de entorno a 0,6 y los de umbral de entorno 0,3.

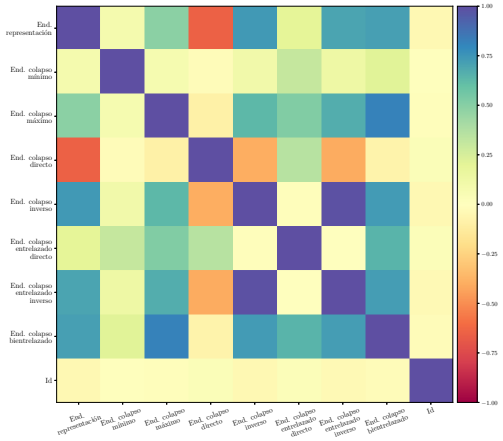
Por ello, para distinguir un conjunto de circuitos de otro puede ser práctico generar los λ-umbrales de distintos tipos de circuitos patrón seleccionados, para luego estimar si el conjunto de circuitos que queremos analizar cumple o no las propiedades de alguno de dichos patrones.

5.5. Correlación entre las distintas métricas

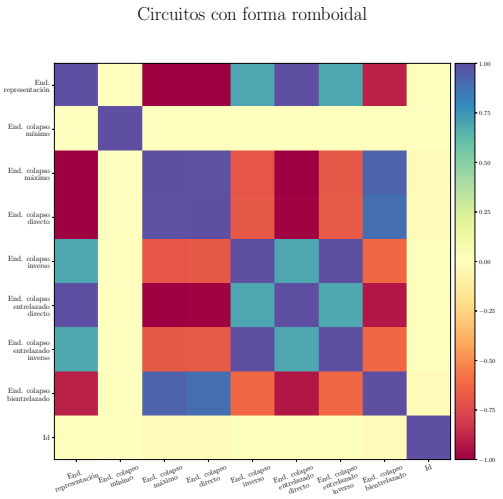
Una vez procesados los datos, trataremos de analizar si existe algún tipo de correlación entre las distintas métricas definidas o si hay algún tipo de relación con la función booleana. Para ello, hemos calculado las siguientes matrices de correlaciones:



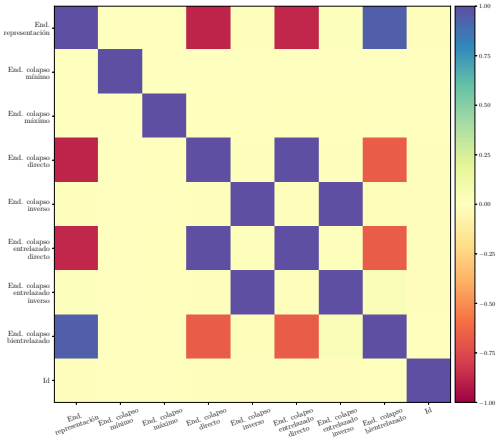
(a) Circuitos mínimos. Ampliar



(b) Gramáticas.Ampliar



(c) Circuitos con forma romboidal. Ampliar



(d) Circuitos con forma 4-romboidal. Ampliar

Figura 5.7: Matriz de correlaciones entre las distintas endogamias.

Además, hemos realizado otra visualización combinando todos los circuitos con los que vamos a trabajar, la cual se puede encontrar aquí. Como es posible ver, en todos los casos hay una clara correlación entre la endogamias por colapso inverso y la endogamia por colapso entrelazado inverso, así como entre las endogamias por colapso directo y colapso entrelazado directo; lo cual hasta cierto punto era previsible, ya que la diferencia entre $|E_l(C)|$ y $2^{n-l} - 1$ en la práctica suele ser nula o

cercana a dicho valor.

Sin embargo, la posible existencia de correlación entre el resto de endogamias es muy difusa, y muy ligada al tipo de circuito involucrado. Nótese que el conjunto de circuitos mínimos con el que se trabaja es de mayor cardinalidad que el resto de los tipos de los circuitos. Por ello, consideramos una muestra suficientemente representativa de la correlación entre endogamias el caso de las gramáticas, ya que la diversidad estructural de circuitos es mayor: los circuitos mínimos tienden a tener baja profundidad y anchura por la propia definición de los mismos, pero sin embargo, las gramáticas son circuitos estructuralmente variados, con un amplio rango de número de puertas empleadas por las mismas. Esta afirmación no se debe interpretar como algo basado en un estudio en profundidad de la densidad de circuitos y su tipo de endogamias en el subconjunto analizado, sino como una razonable e intuitiva aproximación basada en la presunción de distribución equiespaciada de las gramáticas que generan funciones computadas por algún circuito mínimo.

Por último, destaca la nula correlación entre endogamias y funciones booleanas, lo cual era algo desgraciadamente esperable.

5.6. Correlación entre las endogamias y número de puertas empleadas

Por otra parte, una pregunta razonable es ver cuál de estas endogamias refleja mejor el número de puertas empleadas. Para ello, dada la variable aleatoria E asociada a la endogamia de un circuito y la variable aleatoria G asociada al número de puertas, estudiaremos si hay una correlación entre E y G (caso Endogamia-Puertas), si la hay entre $1 - E$ y G (caso Endogamia opuesta - Puertas) o si existe dicha correlación entre $\frac{1}{E}$ y G (caso Endogamia inversa - Puertas).

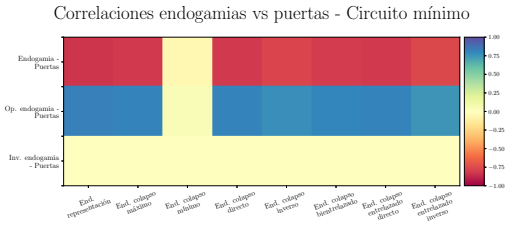


Figura 5.8: Circuitos mínimos. Ampliar

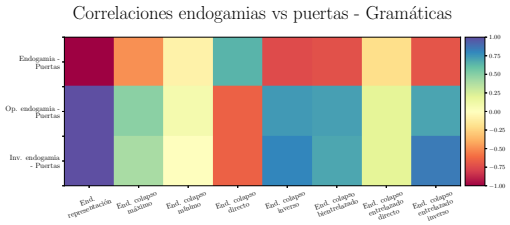


Figura 5.9: Gramáticas. Ampliar

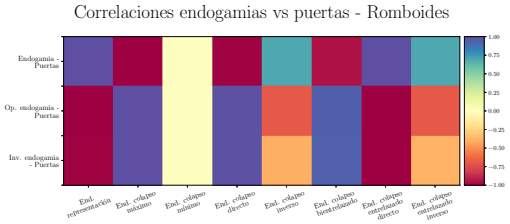


Figura 5.10: Circuitos con forma romboidal. Ampliar

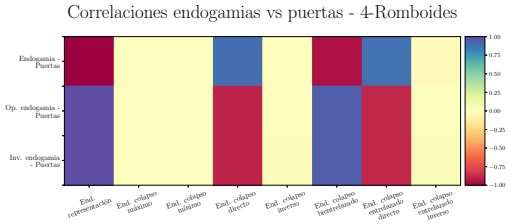


Figura 5.11: Circuitos con forma 4-romboidal. Ampliar

En primer lugar, observemos, como era de esperar, que la endogamia por colapso mínimo no ayuda a estudiar los circuitos ya que, en general, es idénticamente nula. Así, omitiendo en lo que sigue dicha métrica, prosigamos el análisis. Como se puede observar, en los circuitos mínimos hay una correlación positiva en el segundo caso (endogamia opuesta vs número de puertas) y una

correlación negativa en el primero (endogamia vs número de puertas) independientemente del tipo de endogamia. Esto nos lleva a pensar (en contraposición a lo que sucede con las siguientes gráficas) que dado un conjunto de circuitos, un posible test de validación sobre si estos son mínimos o no sería calcular las correlaciones planteadas y observar si dicho conjunto se comporta como en la gráfica expuesta.

Por otra parte, en el caso de las gramáticas obtenemos que la endogamia por colapso entrelazado directo no correlaciona con el número de reglas que esta emplea. Además, la endogamia por colapso directo correlaciona de manera opuesta a como lo hacía en el caso de los circuitos mínimos.

En el tercer caso, observamos que los circuitos romboidales también se comportan de manera distinta a los circuitos mínimos: la endogamia por representación, colapso inverso y colapso entrelazado inverso se comportan a la inversa que en el caso mínimo. Y finalmente, el caso 4-romboidal nos hace ver algo todavía más interesante: no nos son útiles las endogamias por colapso máximo, colapso inverso y colapso entrelazado inverso; y de las que nos son útiles, la endogamia por colapso directo y colapso entrelazado directo correlacionan de manera opuesta al caso mínimo.

Esto tiene su explicación: los circuitos romboidales se han elegido *ad hoc* con alta endogamia en sus niveles inferiores y baja endogamia en sus niveles superiores, y los 4-romboidales no tienen endogamia alguna siquiera en los 3 niveles superiores, lo que conduce a un comportamiento inútil de las endogamias inversas y un comportamiento anómalo de las endogamias directas.

Sin embargo, en esta retahíla de diferencias entre un caso y otro nos damos cuenta de un hecho: la endogamia por colapso bientrelazada siempre se comporta de una manera semejante. Este hecho pone de manifiesto la idea intuitiva que yacía sobre la definición de dicha endogamia: en un circuito, todo depende de las conexiones reales, no del nivel en el que nos encontremos. Por ello, parece razonable afirmar que, de todas las endogamias, la que refleja mejor el número de puertas empleadas es esta.

5.7. Correlación entre distintos datasets

Una vez hecho el análisis anterior, tratemos de intentar hallar algún tipo de relación entre las endogamias, los circuitos y las funciones que computan para distintos tipos de circuitos. De nuevo, trataremos de estudiar la correlación entre distintas variables aleatorias. En este caso, y definiendo E y G como en el apartado anterior, se ha optado por:

1. Reutilización simple: estudiar la correlación entre las variables $X = (1 - E_1) \cdot G_1$, $Y = (1 - E_2) \cdot G_2$.
2. Reutilización compleja: estudiar la correlación entre las variables $X = G_1$, $Y = (1 - E_2) \cdot G_2$.
3. Cociente: estudiar la correlación entre las variables $X = E_1 \cdot G_2$, $Y = E_2 \cdot G_1$.
4. Cociente opuesto: estudiar la correlación entre las variables $X = (1-E_1) \cdot G_2$, $Y = (1-E_2) \cdot G_1$.

Intuitivamente en el primer caso tratamos de comparar el número de puertas empleadas en un tipo de circuitos y su *exogamia* (el valor $1 - E$) con el mismo valor en otro tipo de circuitos, de manera que algún tipo de correlación directa nos quiera decir que a mayor número de puertas menor valor de exogamia, y por tanto, menor valor de endogamia.

La segunda situación es parecida, pero despreciando el término de exogamia en dicha definición. La motivación de dicha decisión es aplicar dicho análisis a circuitos mínimos, donde la endogamia obtenida será, idealmente, la menor posible.

En el tercer caso, tratamos de buscar correlación entre $\frac{E_1}{E_2}$ y $\frac{G_1}{G_2}$. Sin embargo, por cuestiones técnicas de precisión se ha optado por esta transformación aún sin ser equivalente. Es decir, los resultados de este análisis no son directamente extrapolables a la transformación $(\frac{E_1}{E_2}, \frac{G_1}{G_2})$, pero igualmente se ha decidido llevar a cabo. Finalmente, el último análisis se justifica de la misma manera.

Recordemos, antes de presentar el análisis, que se ha decidido para los conjuntos de circuitos con forma romboidal y 4-romboidal que se tomará, como representante de la clase de circuitos que computan la misma función, al circuito de menor endogamia.

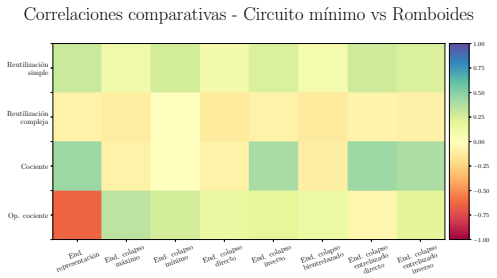


Figura 5.12: Circuitos mínimos frente a circuitos con forma romboidal. Ampliar

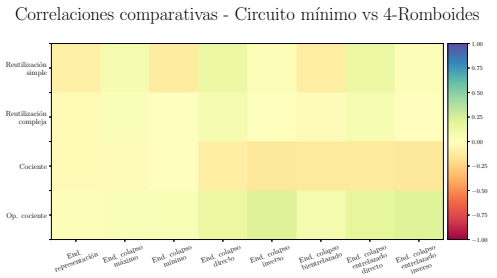


Figura 5.13: Circuitos mínimos frente a circuitos con forma 4-romboidal. Ampliar

Como se puede ver, en ninguno de los dos casos obtenemos algún tipo de relación entre el número de puertas empleado y su endogamia, lo que nos lleva a pensar que este análisis no nos permite diferenciar dos datasets con circuitos distintos ni ayuda en el estudio de las funciones que estos computan. Finalmente, se decidió realizar el mismo análisis confrontando los circuitos mínimos con la gramática asociada a estos.

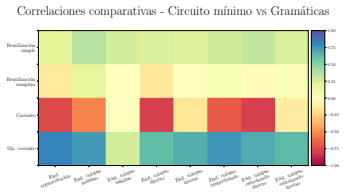


Figura 5.14: Circuitos mínimos frente a gramáticas. Ampliar

En este caso parece haber algún tipo de relación en los dos últimos análisis, por lo que parece razonable estudiar comparativamente las gramáticas frente a otros tipos de circuitos.

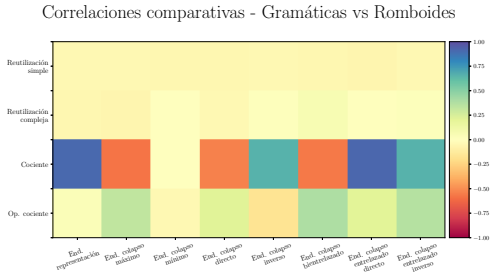


Figura 5.15: Gramáticas frente a circuitos con forma romboidal. Ampliar

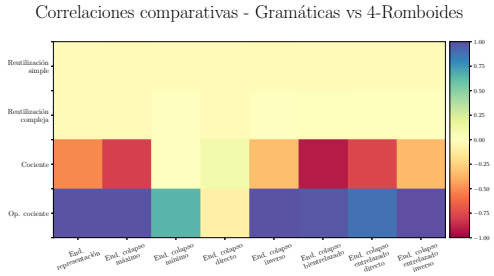


Figura 5.16: Gramáticas frente a circuitos con forma 4-romboidal. Ampliar

Como se puede observar, uno de los invariantes que se preserva en estas tres últimas gráficas es una correlación inversa entre las variables X, Y en el caso cociente para la endogamia por colapso bientrelazado. Junto con la observación de la sección 5.6, en la cual razonábamos que existía una correlación entre el número de puertas y dicha endogamia, esto parece indicar que hay una correlación existente entre el número de reglas de la gramática y el número de puertas que utiliza un circuito vía la transformación de las variables aleatorias realizadas en este caso.

Esta propiedad nos puede servir para poder establecer un posible umbral de minimalidad: en función del número de reglas empleadas por la gramática que compute una función booleana y la endogamia de la misma, podemos establecer unas restricciones necesarias sobre la endogamia por colapso bientrelazada de cualquier circuito que compute dicha función. Entonces, estudiando dicha endogamia en profundidad quizás podamos establecer condiciones necesarias para estimar el mínimo tamaño que debe tener un circuito que compute dicha función.

En cualquier caso, el diferente comportamiento que esta comparativa tiene al enfrentar dos circuitos entre sí frente a enfrentar circuitos y sus gramáticas, nos da que pensar que quizás, y solamente diremos quizás, nos hallemos ante un nuevo e interesante camino de investigación.

Conclusiones

Finalizaremos este trabajo con un breve resumen de todo lo obtenido a lo largo de este documento, y explicaremos una serie de proyectos pendientes que podrían surgir como continuación natural de este experimento.

En primer lugar, recordemos que el objetivo de este experimento era recabar evidencia empírica de la endogamia de los circuitos y estudiar la relación de esta endogamia con la función computada por estos.

Para ello, hemos desarrollado un formalismo matemático muy flexible para definir los circuitos. Este formalismo nos ha permitido de una manera cómoda cambiar el punto de vista entre circuitos y gramáticas allá donde lo hemos necesitado sin mucho esfuerzo, gracias al teorema de equivalencia. Esto nos conduce a la interesante propiedad de que toda la teoría desarrollada para estudiar las gramáticas se puede adaptar de alguna manera para estudiar los circuitos booleanos. Por otra parte, hemos probado que podemos encontrar representantes únicos de nuestros circuitos y dotar al conjunto de los mismos de un orden total gracias a la representación minimal y a la función índice. Esta función índice nos ha servido para desarrollar y demostrar un complejo pero a la vez simple algoritmo con el que recorrer el espacio de circuitos. Complejo por la laboriosidad del mismo y simple por la sencillez de los argumentos empleados en su demostración, los cuales no requieren de vastos conocimientos ni de matemáticas ni de informática.

Sin embargo, la implementación y el desarrollo del mismo han sido particularmente frustrantes por el tiempo invertido en programarlo, testarlo y probar todos los resultados que nos conducen a él; lemas resultantes del estudio bajo prueba y error de las condiciones necesarias para recorrer todos los circuitos. Otro obstáculo encontrado en dicho algoritmo ha residido en la inmensidad del espacio estudiado, el cual computacionalmente era inabarcable para el portatil con el que se ha trabajado. Esta inmensidad se reflejó no sólo a la hora de recorrer el espacio, sino a la hora de procesar los datos que en su recorrido obteníamos: todas las decisiones de restringir el conjunto de datos sobre el que analizar nuestros datos (como en el caso de los circuitos romboidales o los 4-romboidales) han sido por la imposibilidad de procesar tal cantidad de información.

A mayores, y para poder utilizar los datos recabados, hemos definido una serie de métricas de endogamia y el concepto de λ-umbral con el objetivo de encontrar algún tipo de relación entre los circuitos y las funciones booleanas. Estos conceptos a priori parecían prometedores ya que condensaban de manera adecuada la estructura de un circuito en un valor numérico. Sin embargo, como se ha visto en el capítulo 5, los resultados han sido un poco infructuosos: en general, entre muchas de las métricas existe una correlación y en múltiples ocasiones esto era lo previsible debido a las si-

militudes inherentes en la definición. Y aquellas correlaciones no previstas, aún siendo interesantes, destacan por su escasez. Además, los distintos análisis planteados no han garantizado ningún tipo de certidumbre, sólo meras sospechas.

Por otra parte, se abre de manera insospechada un posible camino de estudio: el estudio de la correlación vía endogamia por colapso bientrelazado entre gramáticas y circuitos; algo que no esperábamos encontrar pero que destaca de manera notable frente al resto de resultados.

Por otro lado, si echamos la vista atrás y nos fijamos en el concepto de grafo and-inversor introducido en las primeras páginas de este trabajo, observamos que una implementación de circuito utilizando este patrón nos garantiza una mayor eficiencia. En vez de requerir $2m$ inputs para construir nuestros circuitos, exigimos trabajar con 4 puertas de dos inputs y 2 puertas de solo un input. Así, el cardinal de $\mathcal{C}_n$ es $2^n \cdot m^{2^n}$, el cual, confrontado a los $(2m)^{2^n}$ circuitos generados con nuestra definición, conlleva una mejora de $2^{2^n-n} \in \mathcal{O}(2^{2^n})$.

De hecho, se puede comprobar que tanto la función índice como el algoritmo del índice permanecen prácticamente impasibles a esta decisión: todos los lemas son fácilmente adaptables a esta implementación. De hacerse así, la cantidad de circuitos redundantes obtenidos se reduce drásticamente, con lo que se podrían obtener mayor cantidad de datos con los que validar o refutar las conjeturas propuestas.

Por último, destaquemos que aunque parezca infructuoso lo logrado con este experimento, no es así: en el proceso de estudiar este espacio se han obtenido resultados interesantes y se plantean nuevas vías de estudio con fuerte carga matemática e informática que a priori son un poco antiintuitivas. Más aún, se ha visto cómo es el proceso de investigación desde su faceta más directa y más sincera, y eso no se aprende en un libro.

Conclusions

This chapter will briefly summarize all the results obtained during our experiment, and it will also include a discussion of the suitability of some future developable projects from this work.

First of all, let us not forget that the objective of this experiment was to collect empiric evidence about the endogamy of the boolean circuits and studying the relation between the endogamy and the boolean functions they compute.

For this purpose, we have developed a mathematical formalism to define our circuits. This formalism has allowed us to exchange circuits and grammars wherever it was needed thanks to the equivalence theorem. That leads us to think that every theory involving grammars can be adapted for studying boolean circuits. Moreover, we have found a unique description of our circuits, their minimal representation. Not only it has been defined a total order on the set of circuits, but also an index function which has enabled us to develop and prove a complex, yet simple, sweeping algorithm of the set of boolean circuits.

Nevertheless, the time-expensive implementation of this algorithm has carried some frustration, due to testing and proving its correction. In addition, other obstacle found in our algorithm was the immense space studied, unmanageable for the used laptop. This immensity reflected both on sweeping the circuits' space and processing all the data collected: every datasets reduction is owed to this reason.

Additionally, we have defined some endogamy metrics and the concept of λ-threshold in order to find any kind of relation between boolean circuits and the functions they compute. Beforehand, these definitions seemed promising because they summarized adequately the structure of a circuit in a numeric value. Nevertheless, as we have seen in Chapter 5, the results obtained were a bit fruitless: between many metrics there exists a trivial correlation, and the analysis of the data does not give a clear answer of the endogamy problem. Accidentally we have obtained a new research path: the study of the correlation between grammars and circuits by using the biintertwined collapse endogamy. This is something we did not expect to find, but we are definitely glad to have done so.

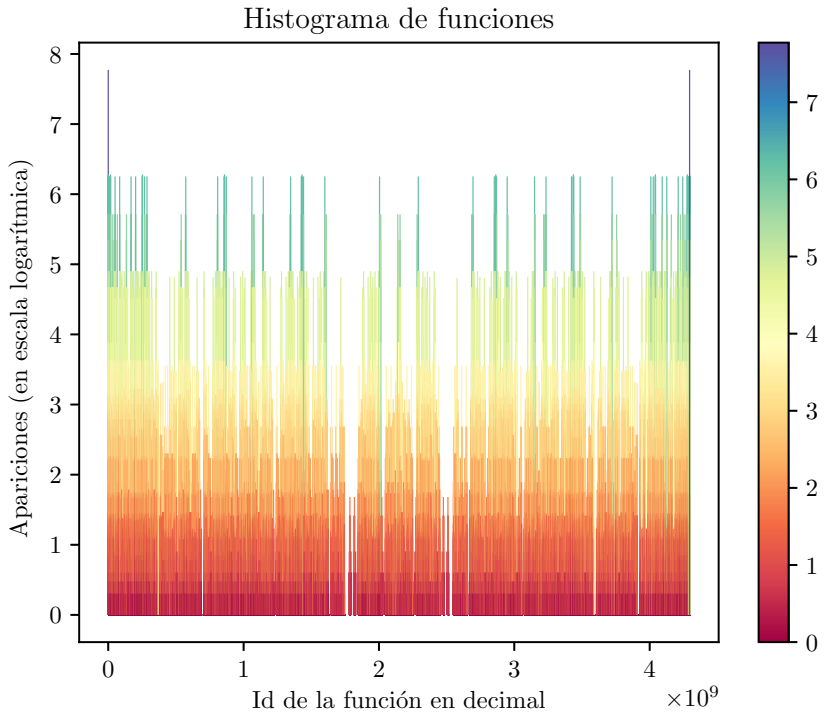
Furthermore, returning the focus to the and-inverter graph mentioned in the introduction of this work, we realize that an implementation of a circuit using this pattern shows a better efficiency: instead of using $2m$ inputs, we need to use twice different gates as we use now. Doing so, the cardinality of $\mathcal{C}$ is $2^n m^{2^n}$ instead of $(2m)^{2^n}$, so the gain resulting is $2^{2^n-n} \in \mathcal{O}(2^{2^n})$. In fact, it can be proven that both the index function and the index algorithm remain almost identical after this change in the implementation. In this way it may be possible to obtain more irredundant data to

validate or refuse our conjectures.

Finally, we want to remark that the great achievement of this project is the path followed in its process. We have opened a path which is likely to be enhanced, a path where mathematics and computer science intertwine. Through the process we have acquired investigation skills ungraspable with only the theoretical content of a book. To end up, we would like to highlight the importance of the results not only for themselves but for the wide knowledge and investigation opportunities that have been brought up.

Anexos

Anexo I: Gráficas de la sección 5.3



Porcentaje de funciones analizadas: 0.016%

Figura 5.17: Histograma de funciones computadas (casos completos). Volver

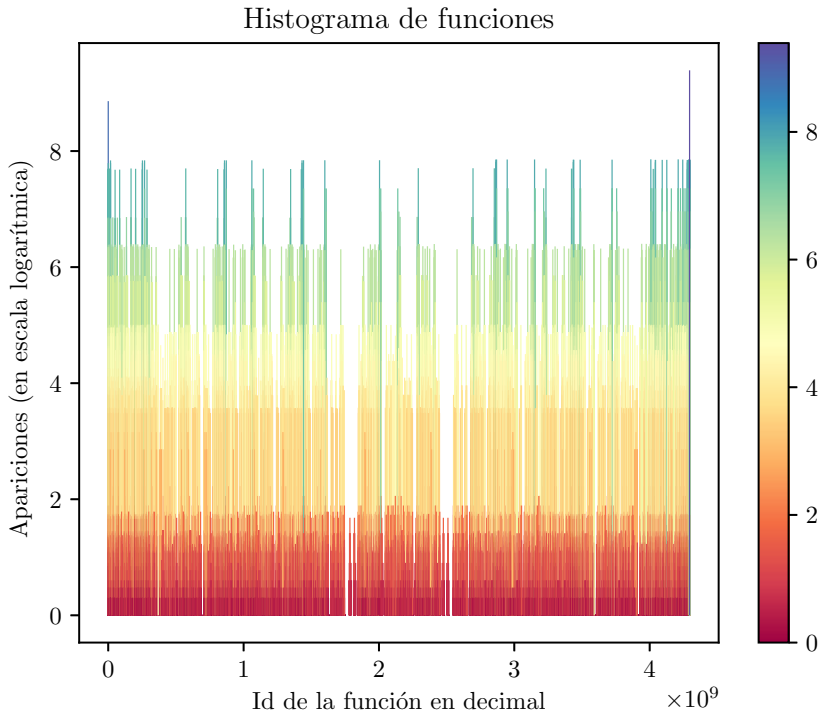


Figura 5.18: Histograma de funciones computadas (casos completos e incompletos). Volver

Anexo II: Gráficas de la sección 5.4

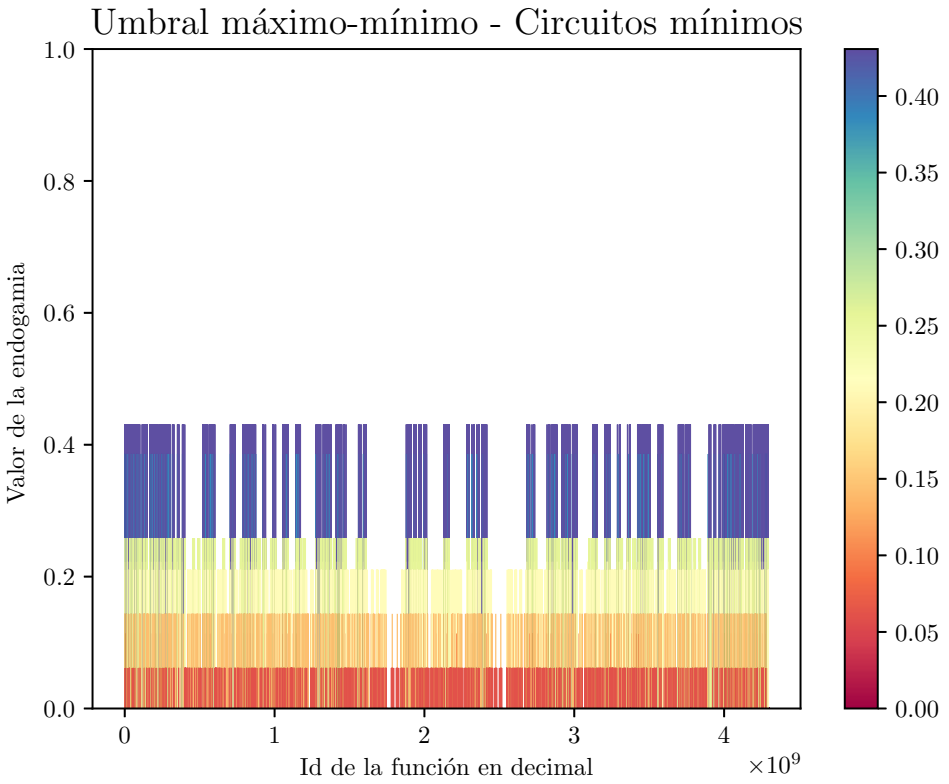


Figura 5.19: λ-umbral máximo-mínimo no entrelazado para circuitos mínimos. Volver

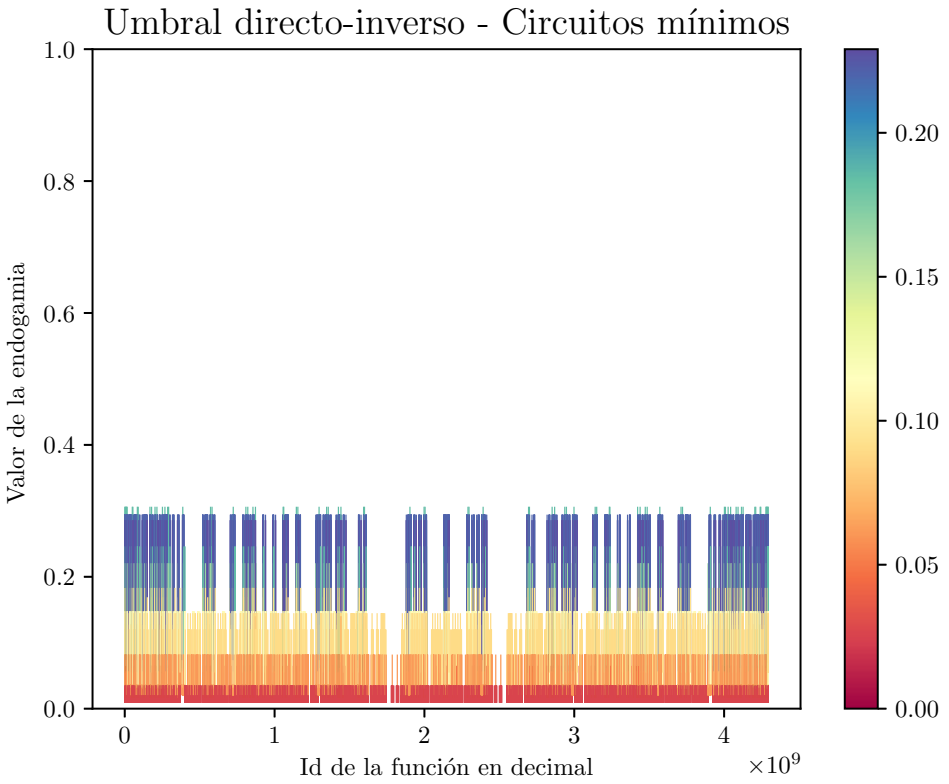


Figura 5.20: λ-umbral directo-inverso no entrelazado para circuitos mínimos. Volver

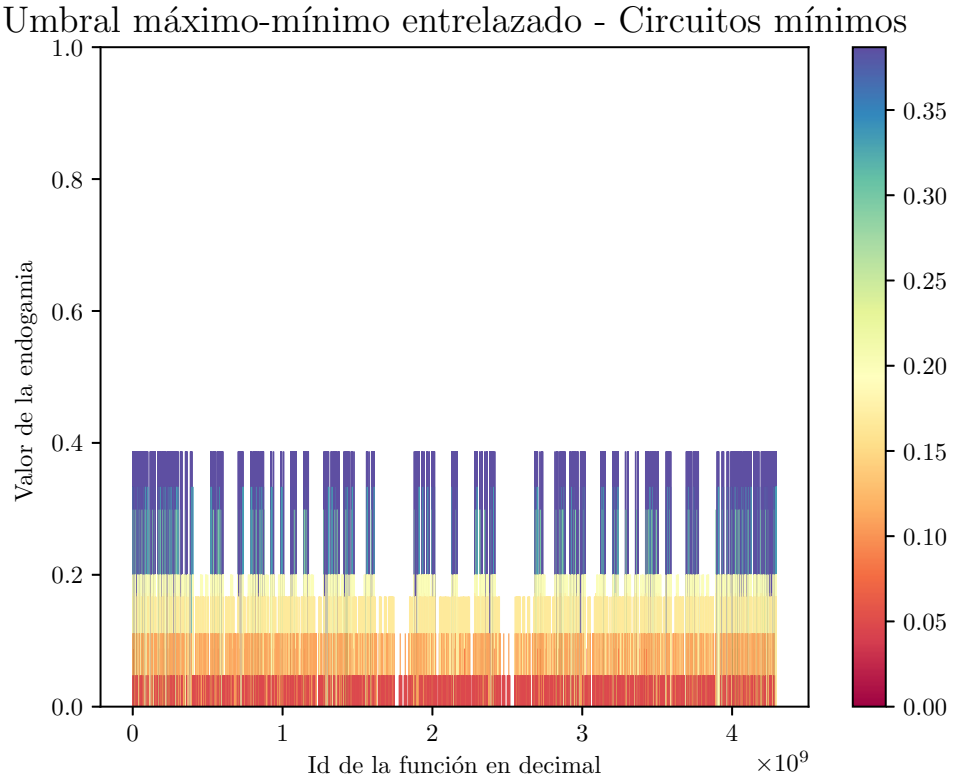


Figura 5.21: λ-umbral máximo-mínimo entrelazado para circuitos mínimos. Volver

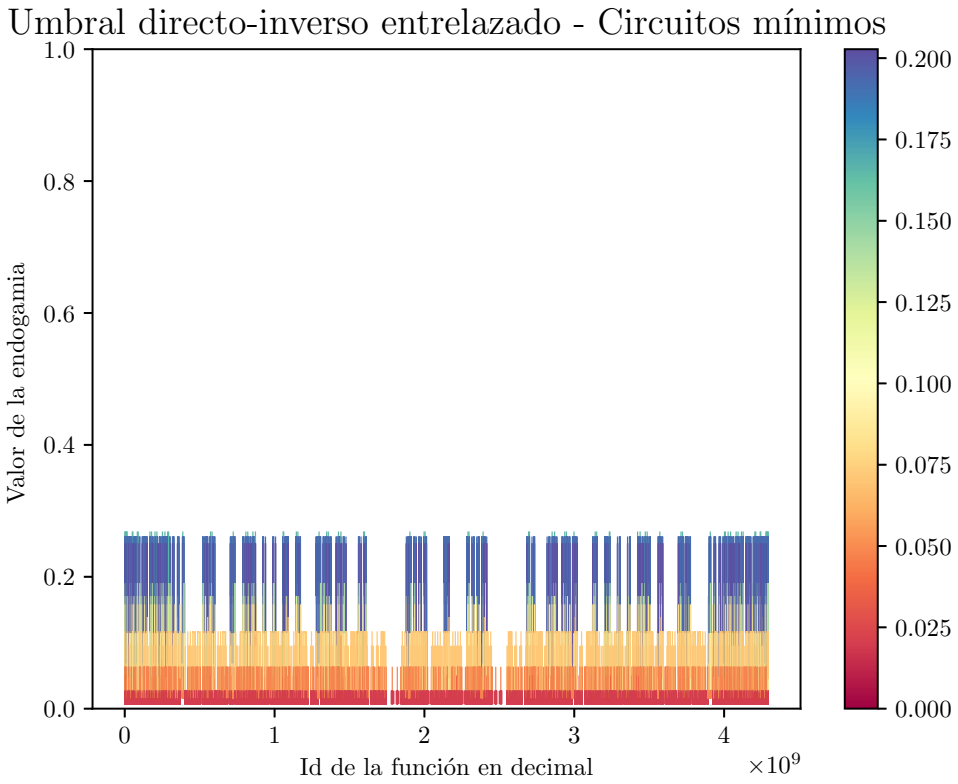


Figura 5.22: λ-umbral directo-inverso entrelazado para circuitos mínimos. Volver

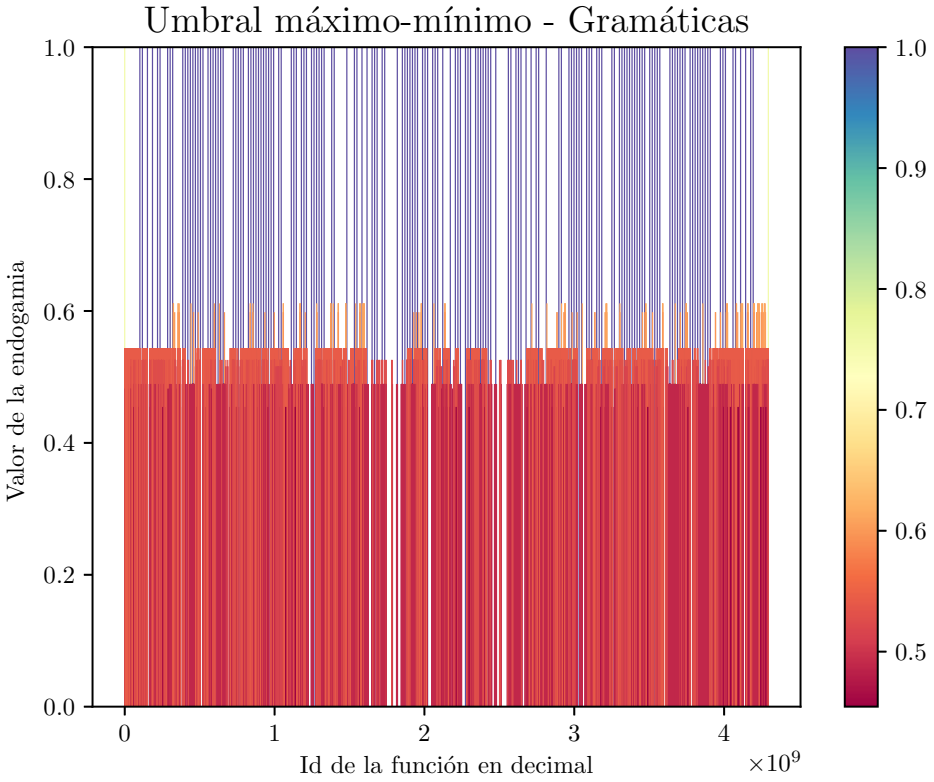


Figura 5.23: λ-umbral máximo-mínimo no entrelazado para gramáticas. Volver

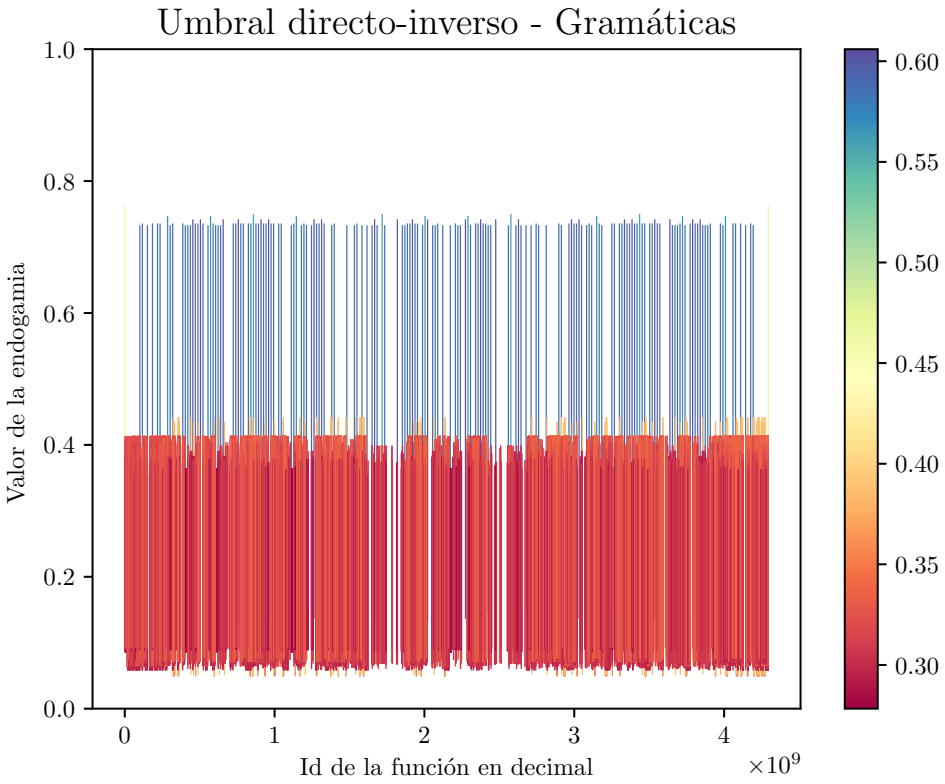


Figura 5.24: λ-umbral directo-inverso no entrelazado para gramáticas. Volver

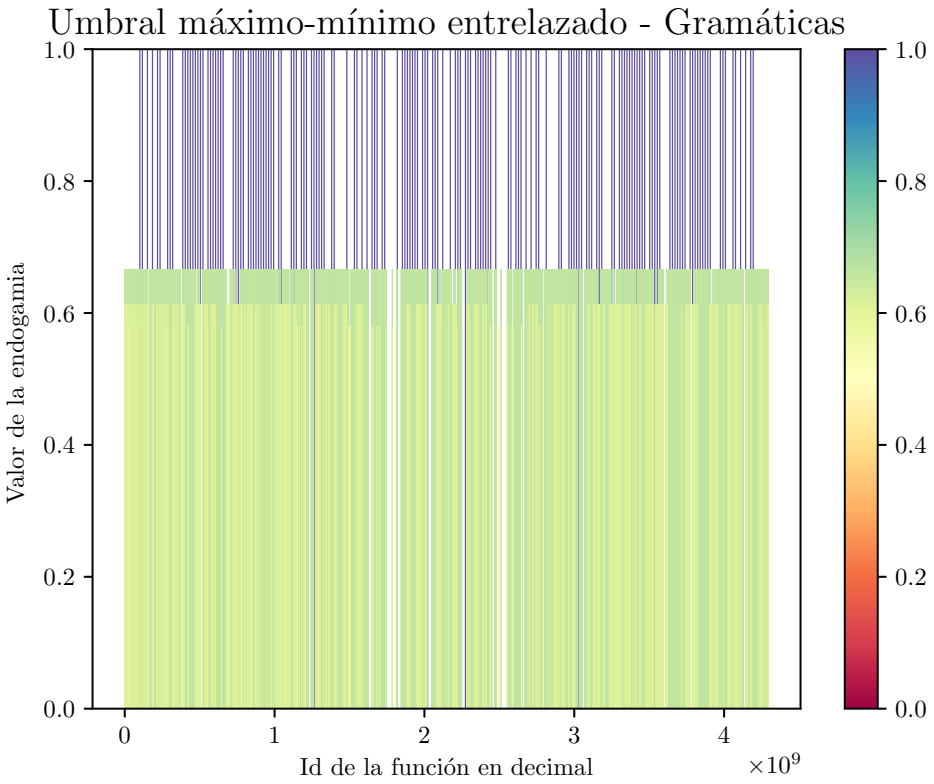


Figura 5.25: λ-umbral máximo-mínimo entrelazado para gramáticas. Volver

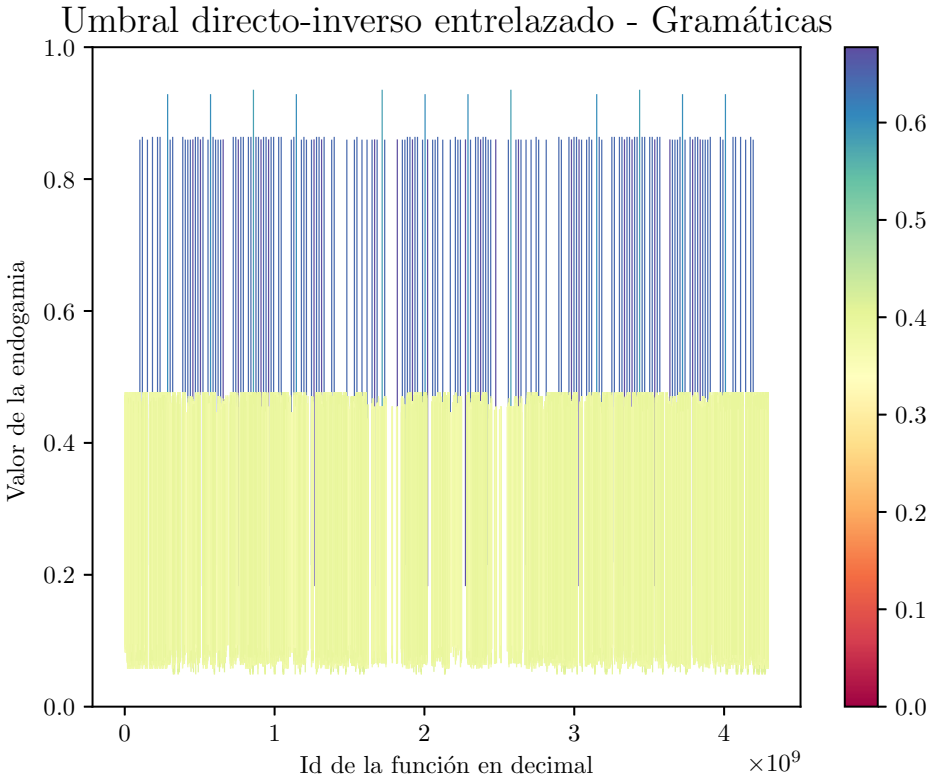


Figura 5.26: λ-umbral directo-inverso entrelazado para gramáticas. Volver

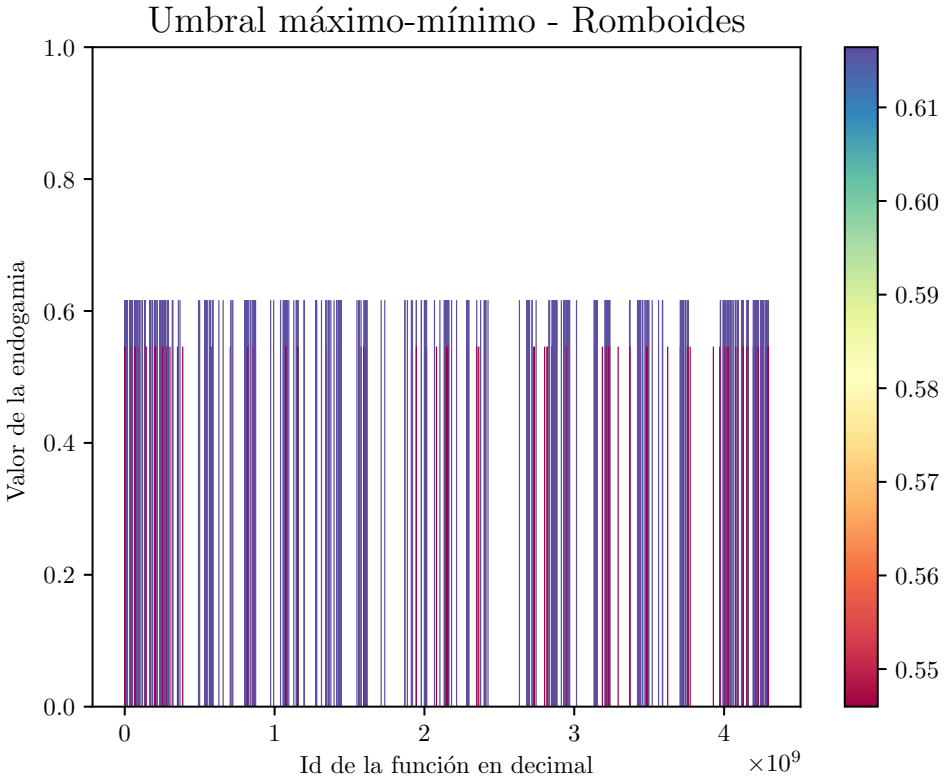


Figura 5.27: λ-umbral máximo-mínimo no entrelazado para romboides. Volver

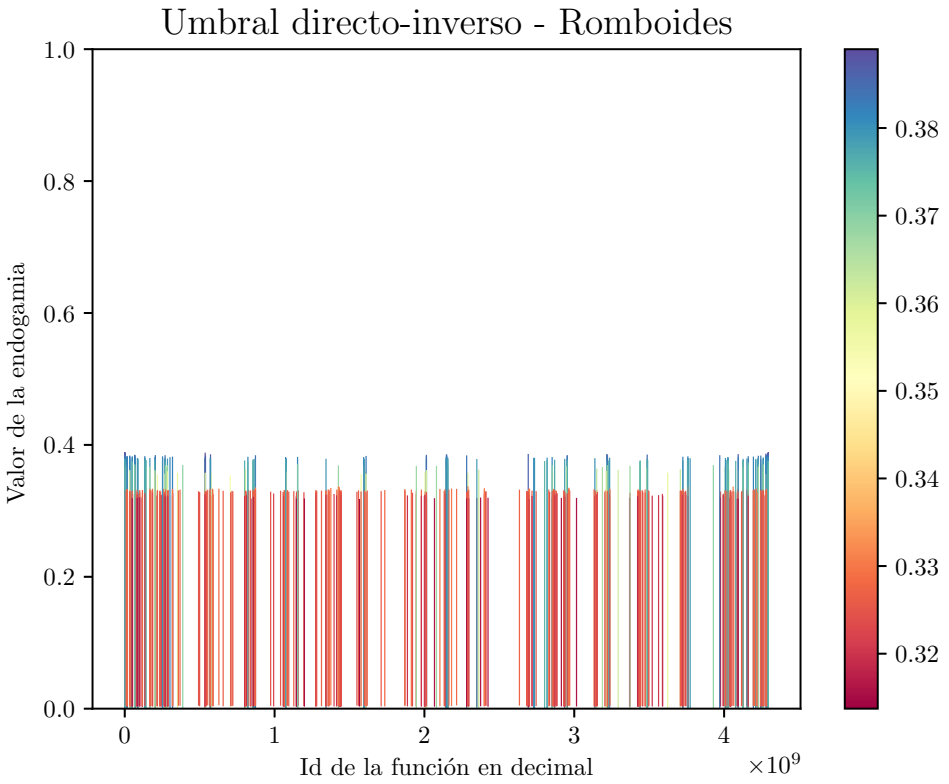


Figura 5.28: λ-umbral directo-inverso no entrelazado para romboides. Volver

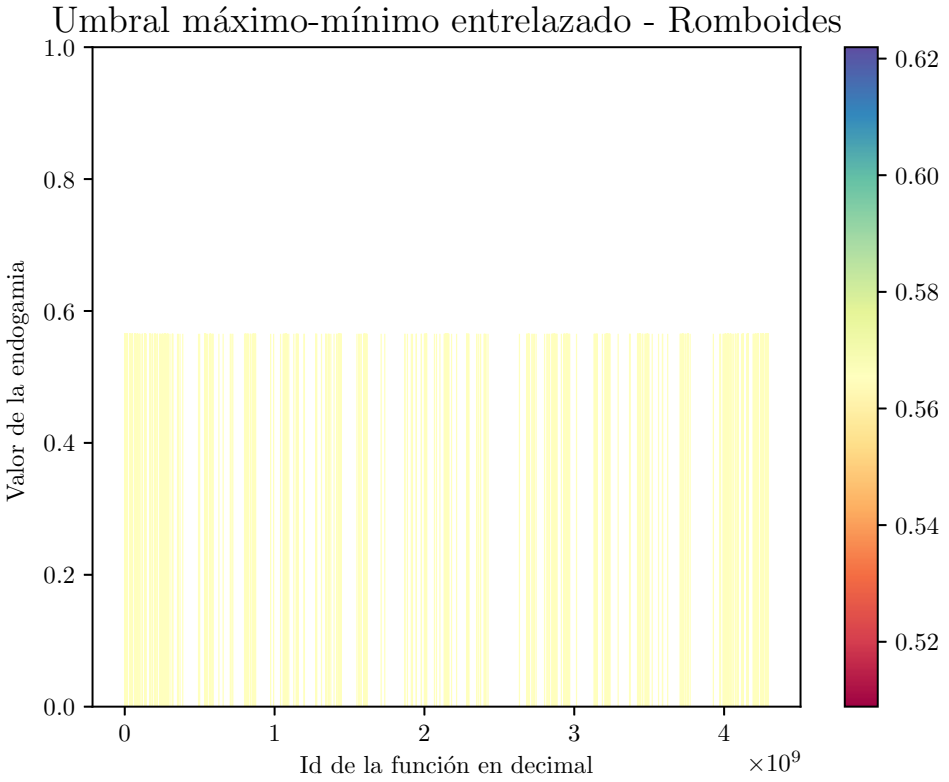


Figura 5.29: λ-umbral máximo-mínimo entrelazado para romboides. Volver

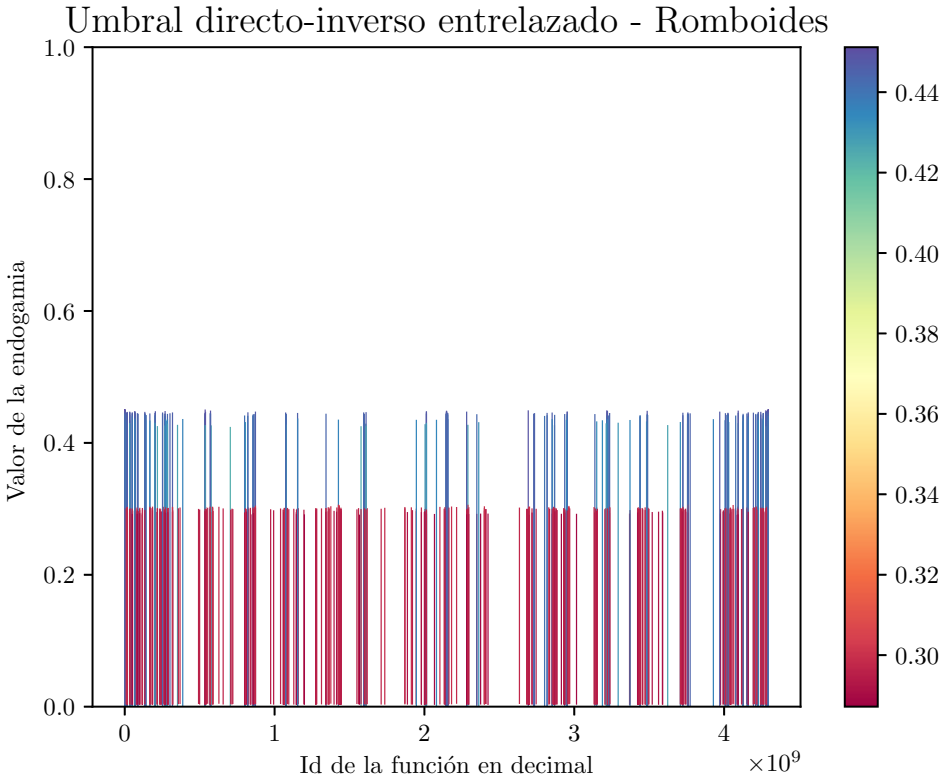


Figura 5.30: λ-umbral directo-inverso entrelazado para romboides. Volver

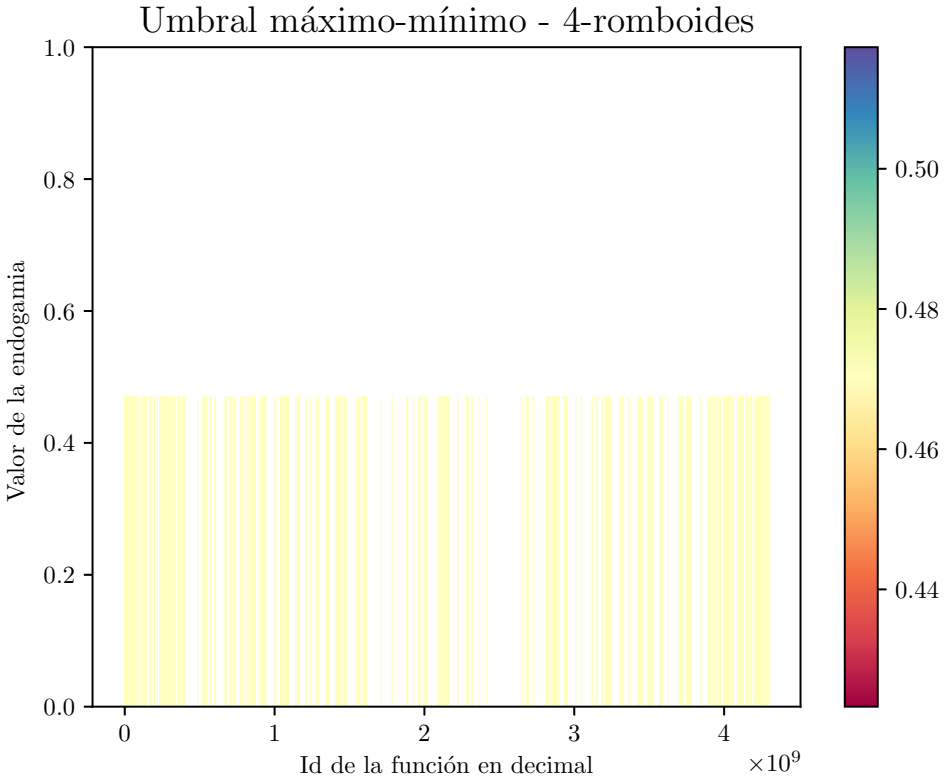


Figura 5.31: λ-umbral máximo-mínimo no entrelazado para 4-romboides. Volver

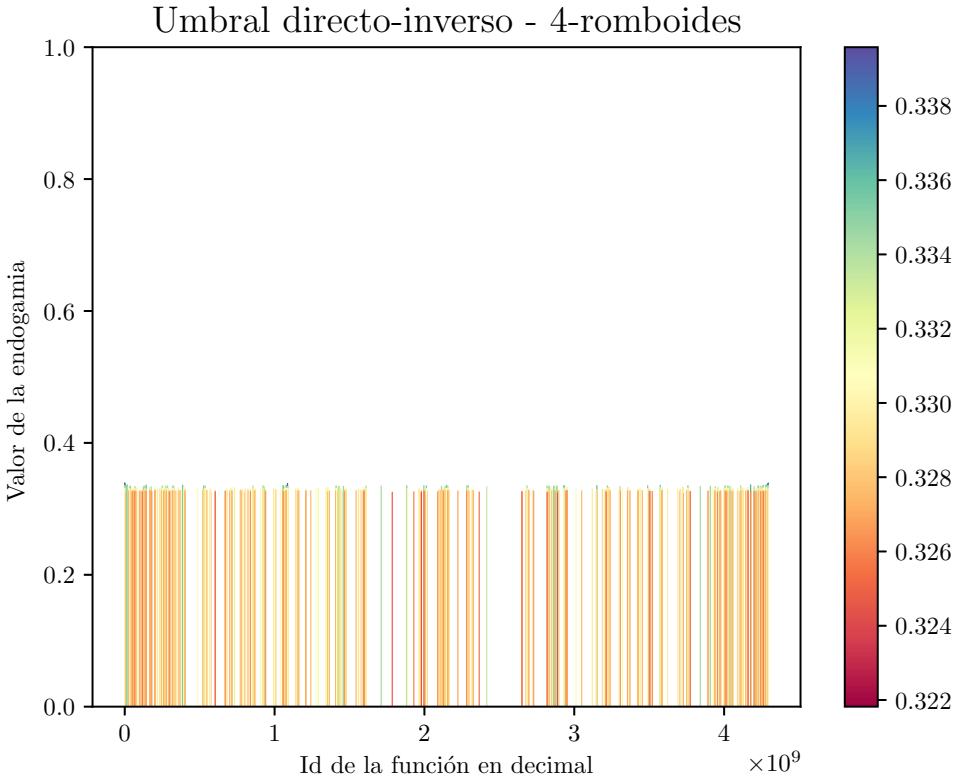


Figura 5.32: λ-umbral directo-inverso no entrelazado para 4-romboides. Volver

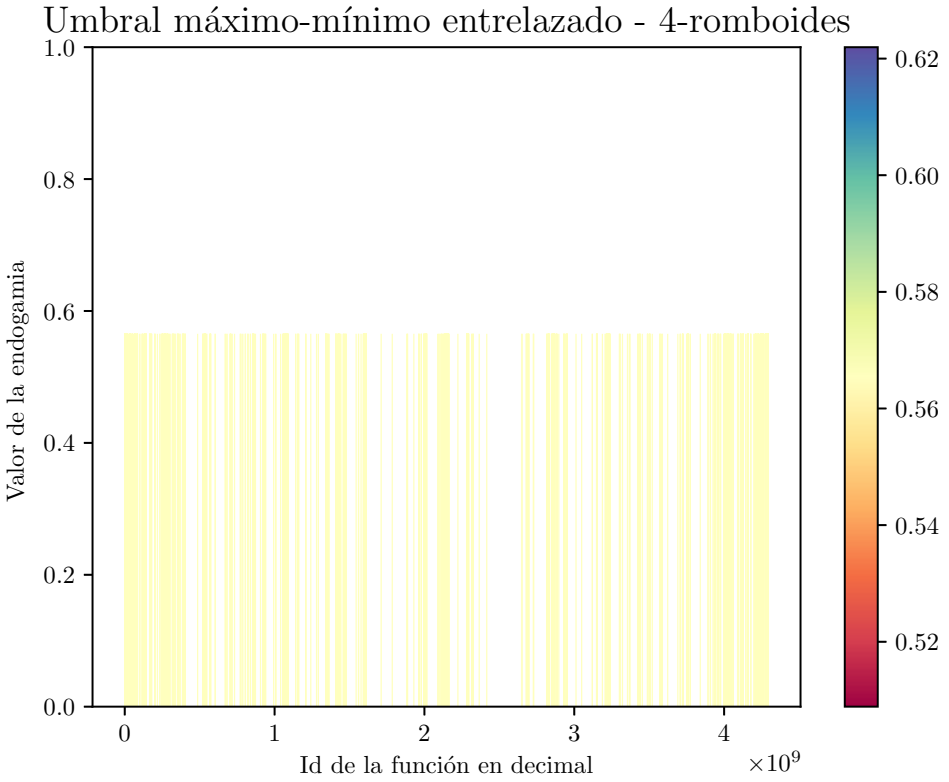


Figura 5.33: λ-umbral máximo-mínimo entrelazado para 4-romboides. Volver

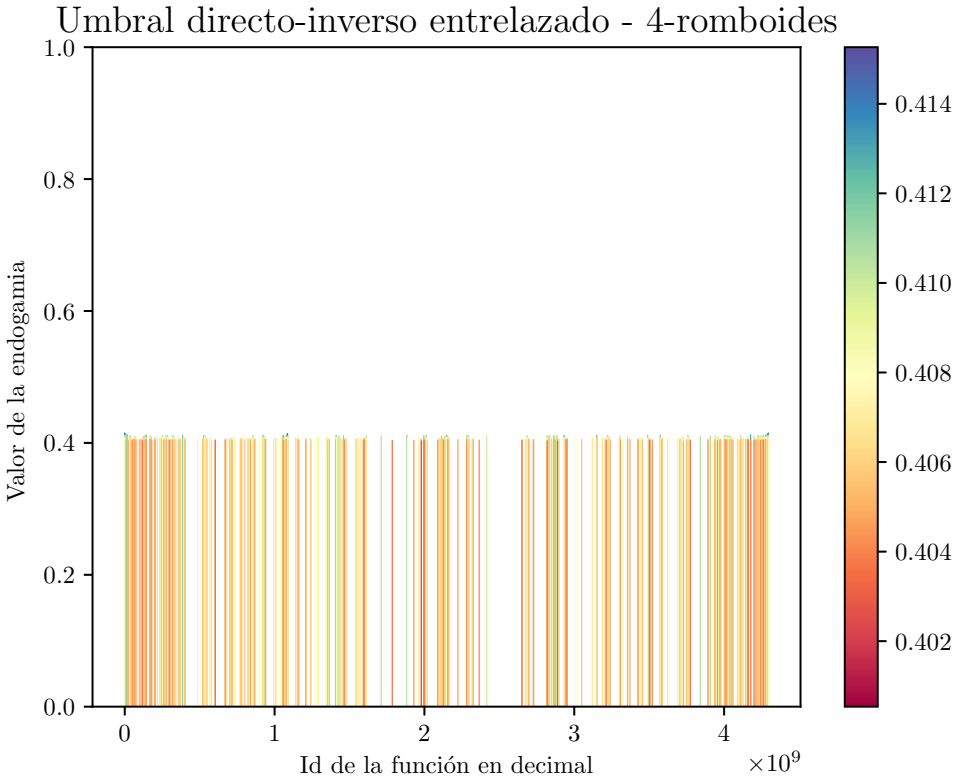


Figura 5.34: λ-umbral directo-inverso entrelazado para 4-romboides. Volver

Anexo III: Gráficas de la sección 5.5

Circuitos con mínimo número de puertas.

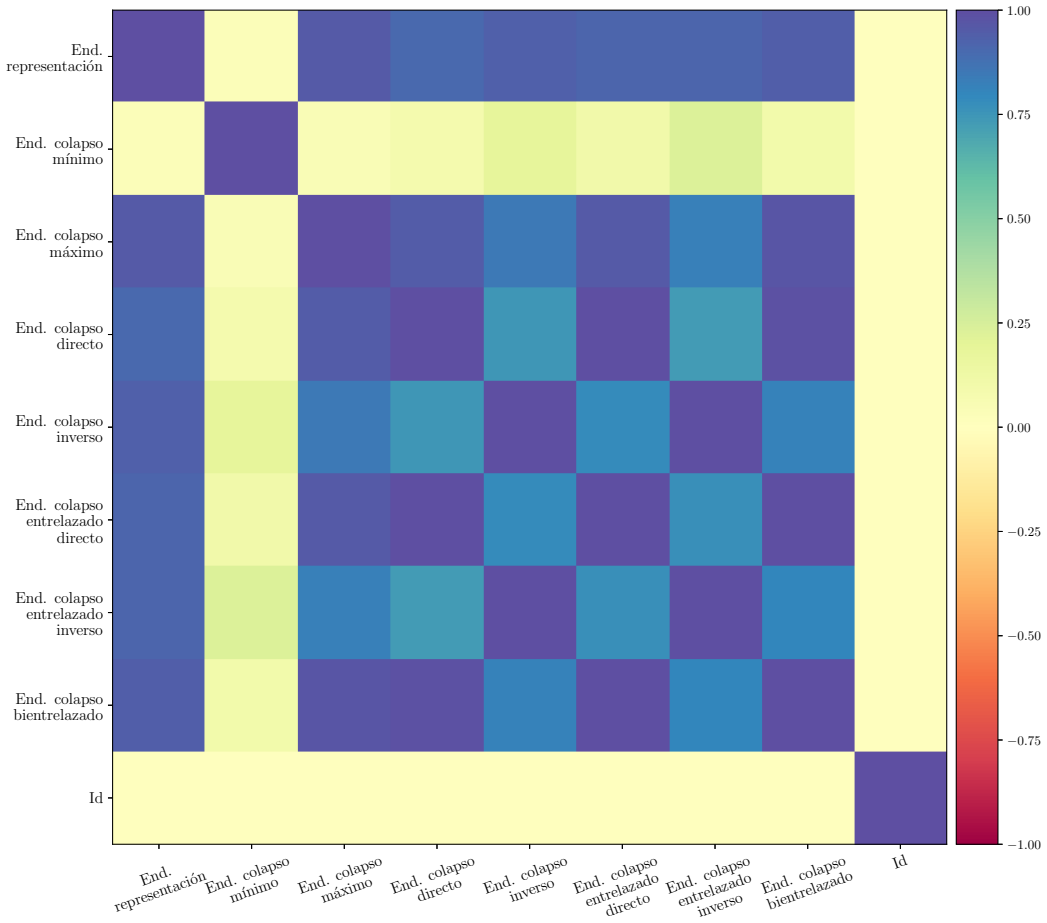


Figura 5.35: Circuitos mínimos. Volver

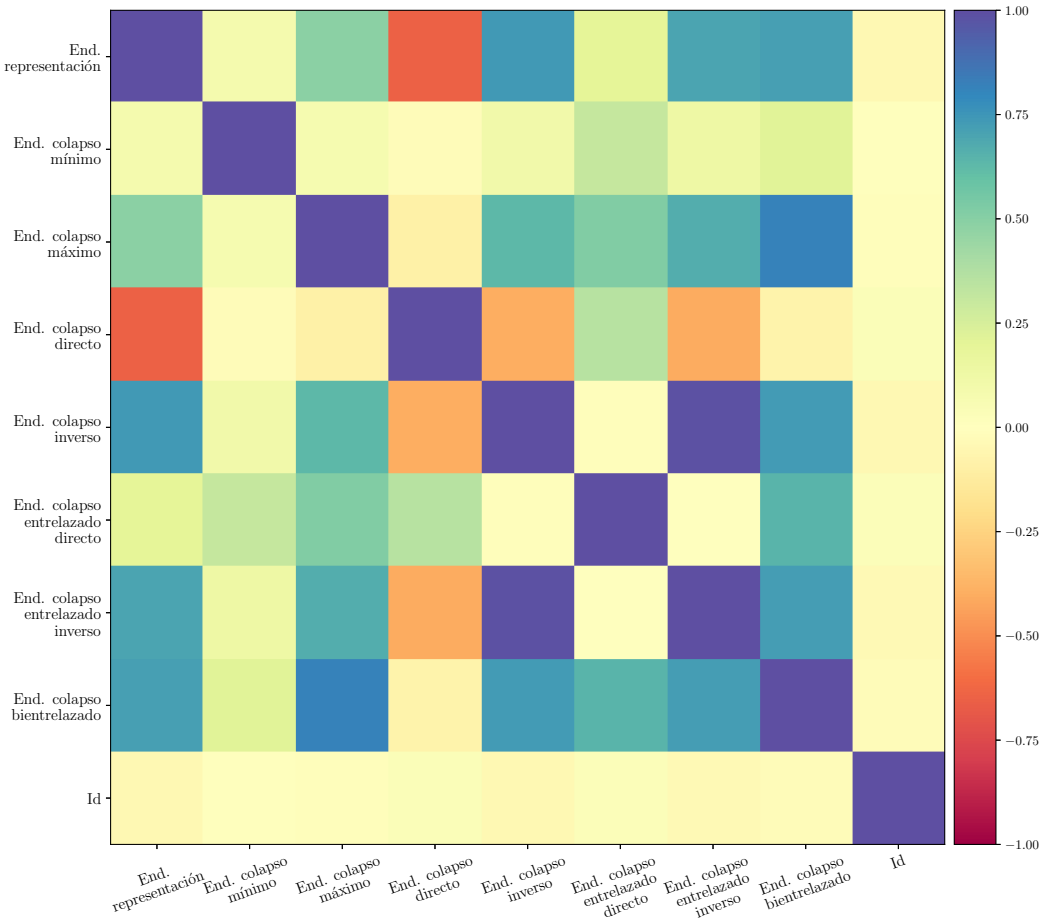


Figura 5.36: Gramáticas. Volver

Circuitos con forma romboidal

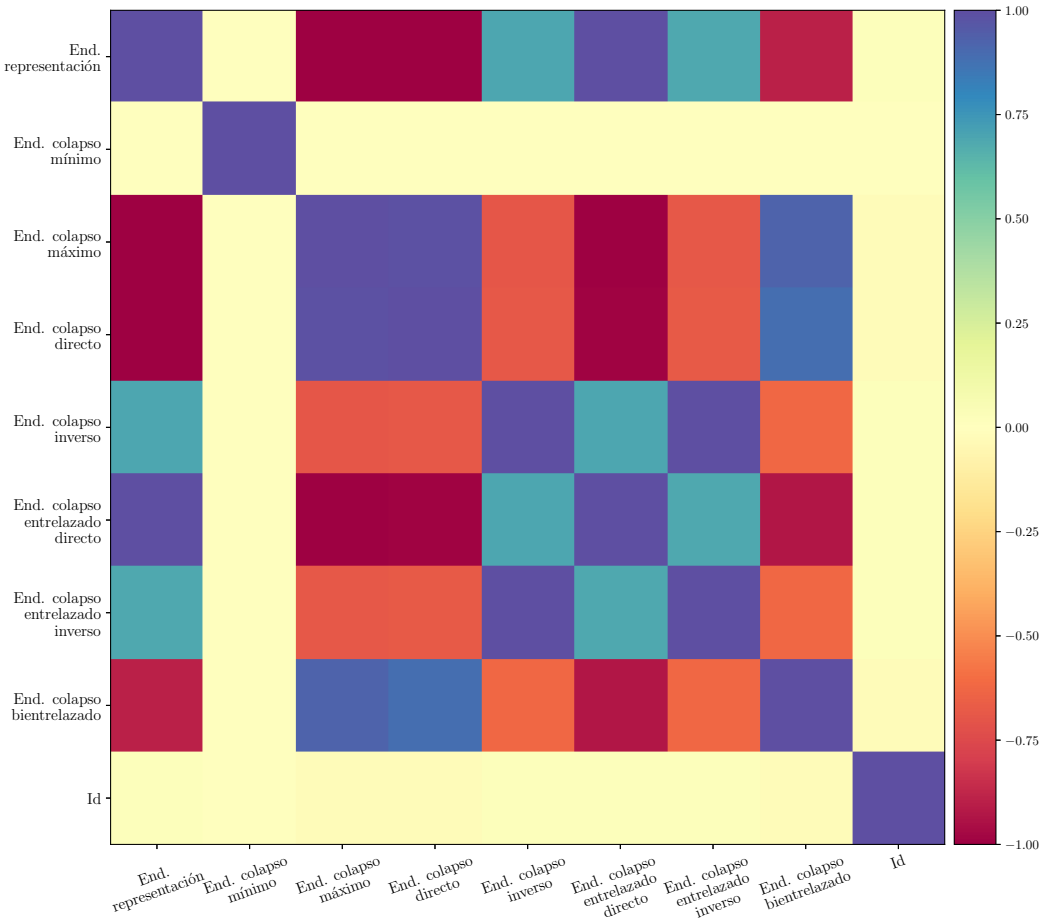


Figura 5.37: Circuitos con forma romboidal. Volver

Circuitos con forma 4-romboidal

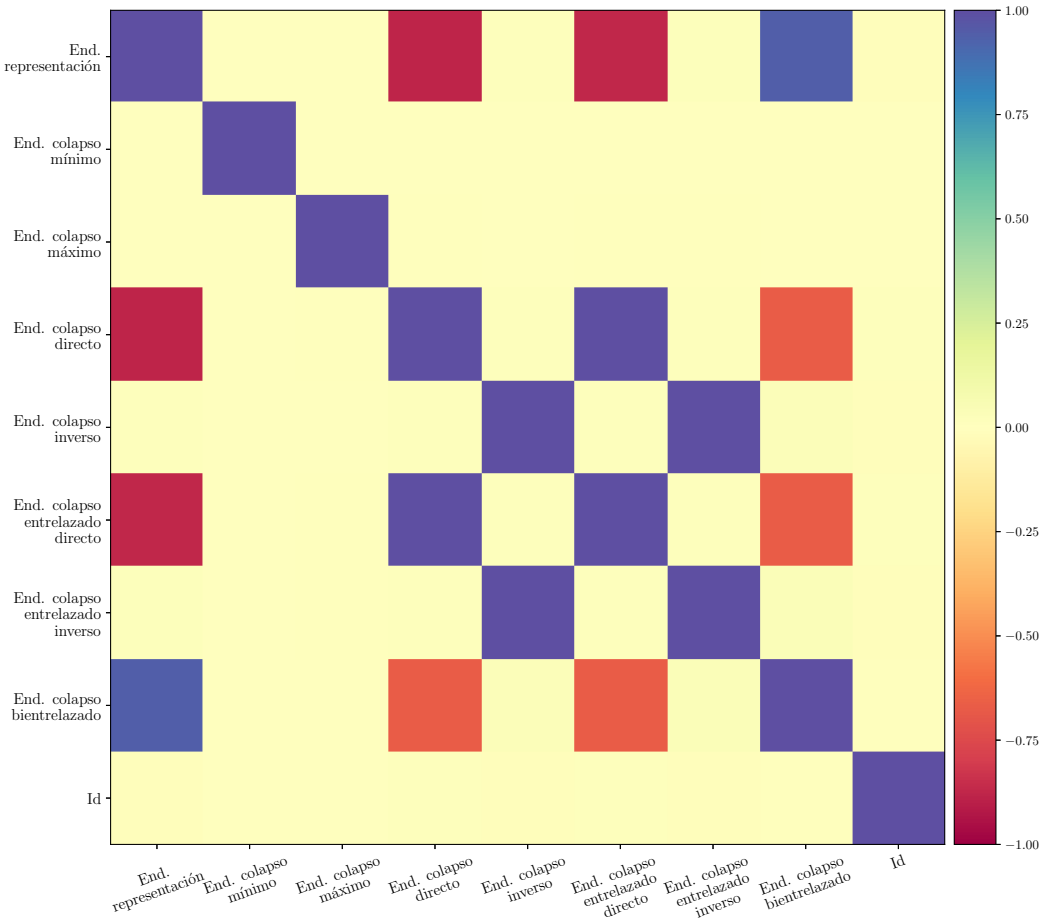


Figura 5.38: Circuitos con forma 4-romboidal. Volver

Todos los circuitos

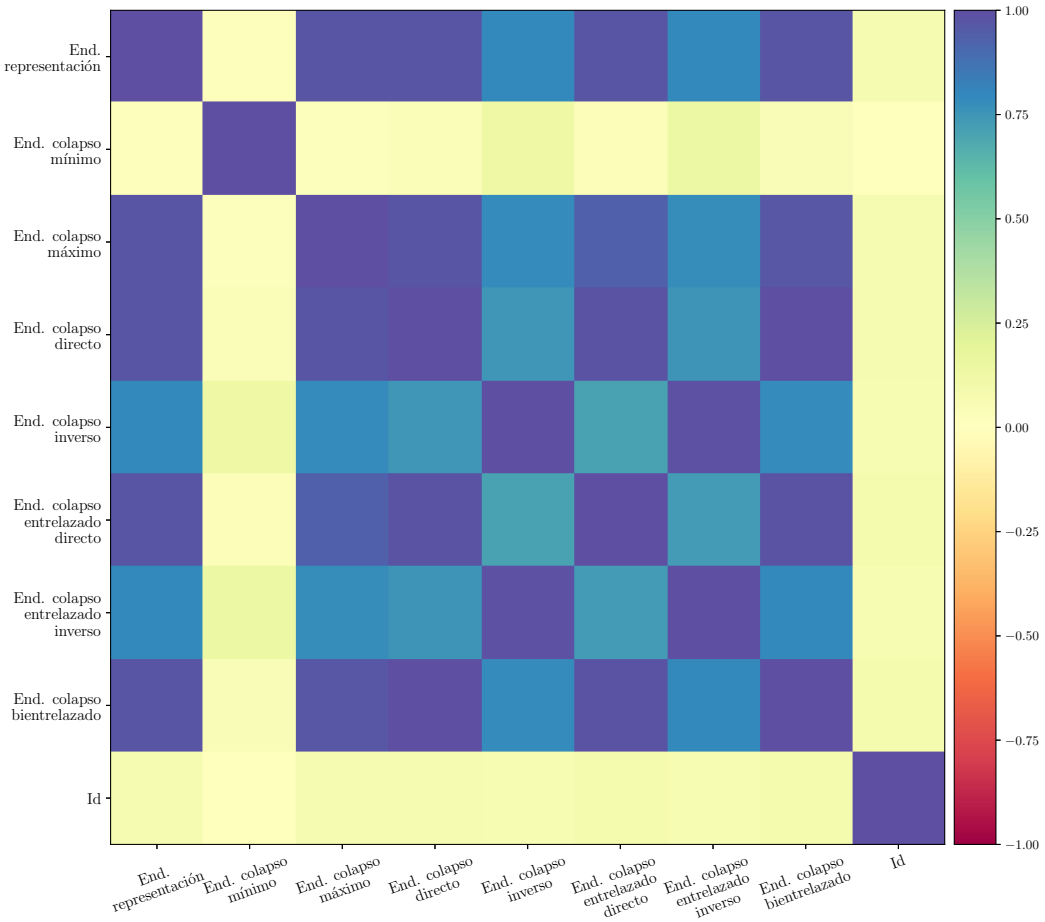


Figura 5.39: Todos los circuitos utilizados. Volver

Anexo IV: Gráficas de la sección 5.6

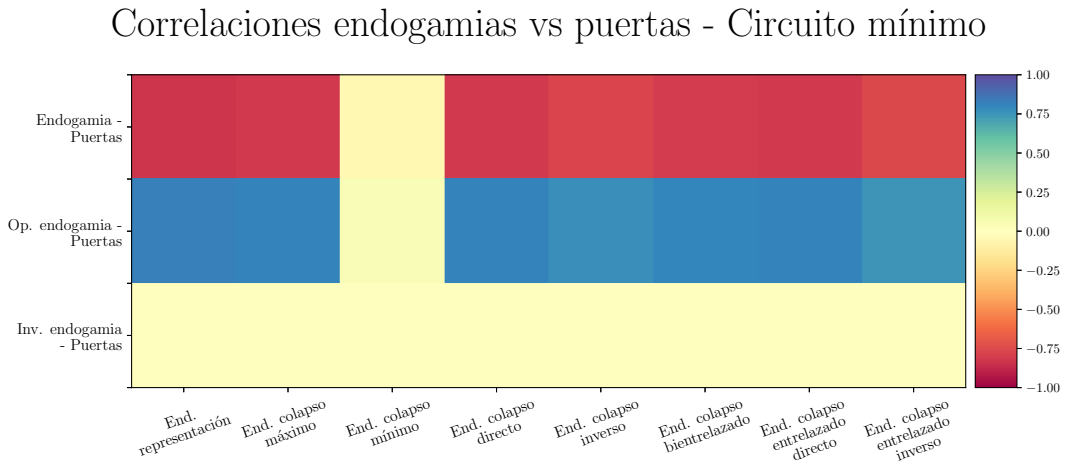


Figura 5.40: Circuitos mínimos. Volver

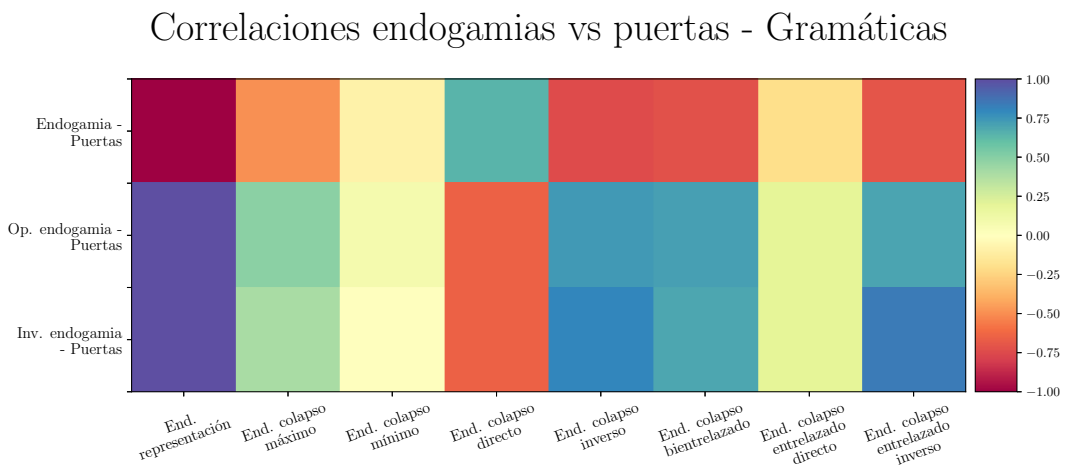


Figura 5.41: Gramáticas. Volver

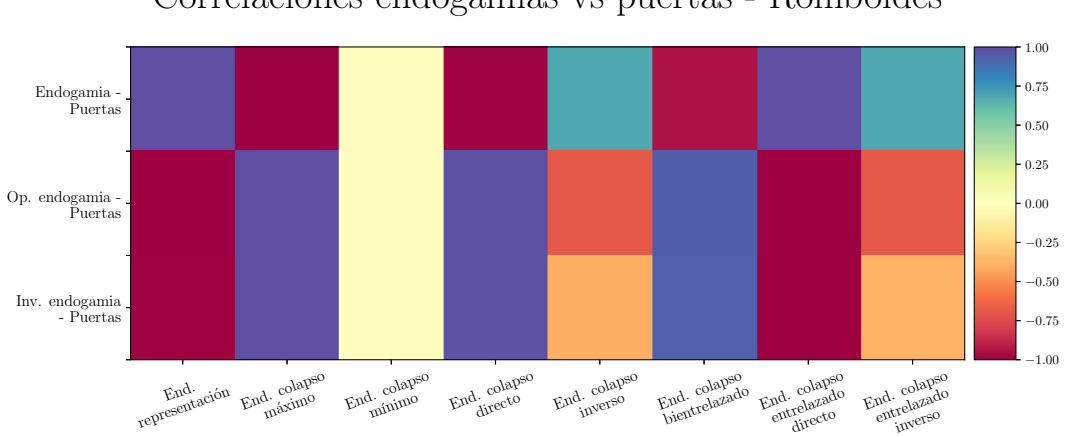


Figura 5.42: Circuitos con forma romboidal. Volver

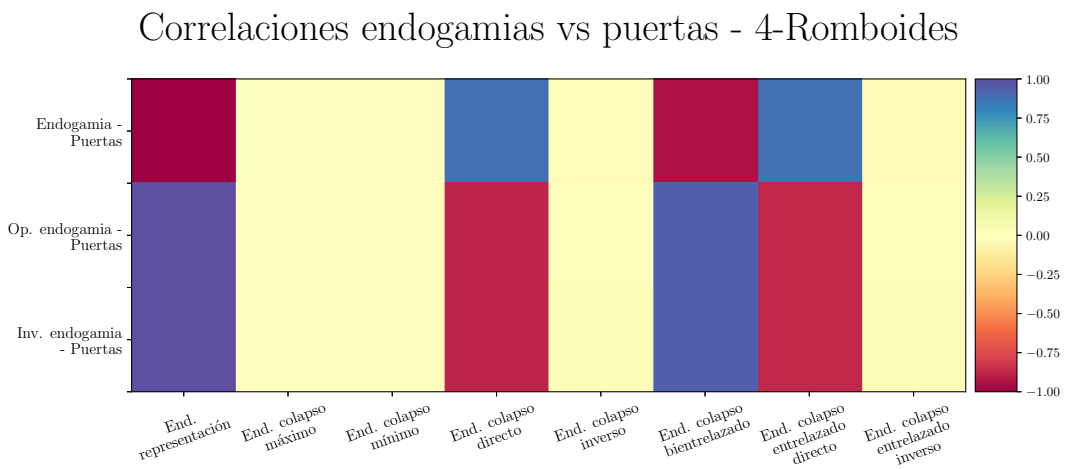


Figura 5.43: Circuitos con forma 4-romboidal. Volver

Anexo V: Gráficas de la sección 5.7

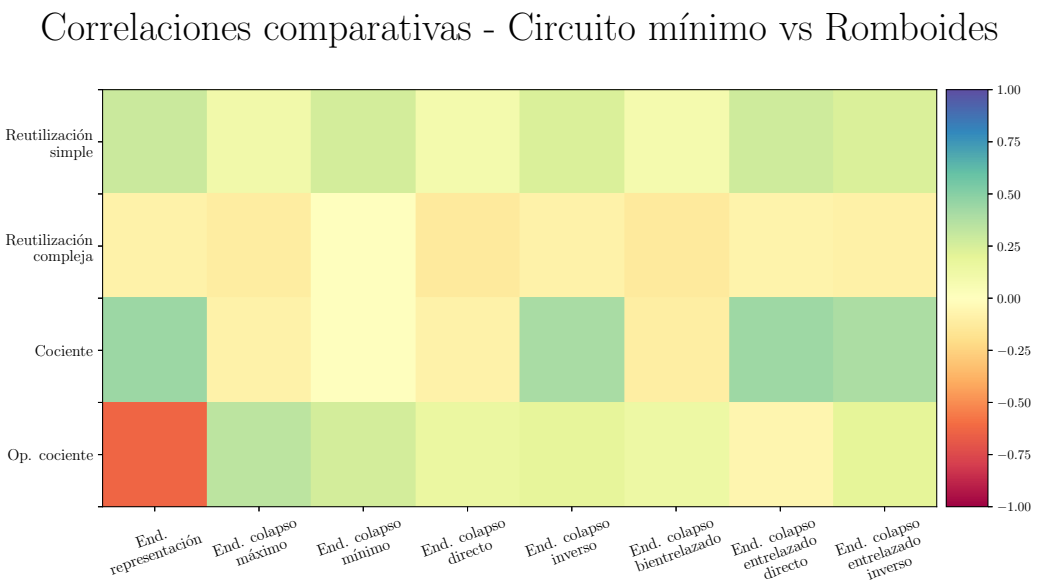


Figura 5.44: Circuitos mínimos frente a circuitos con forma romboidal. Volver

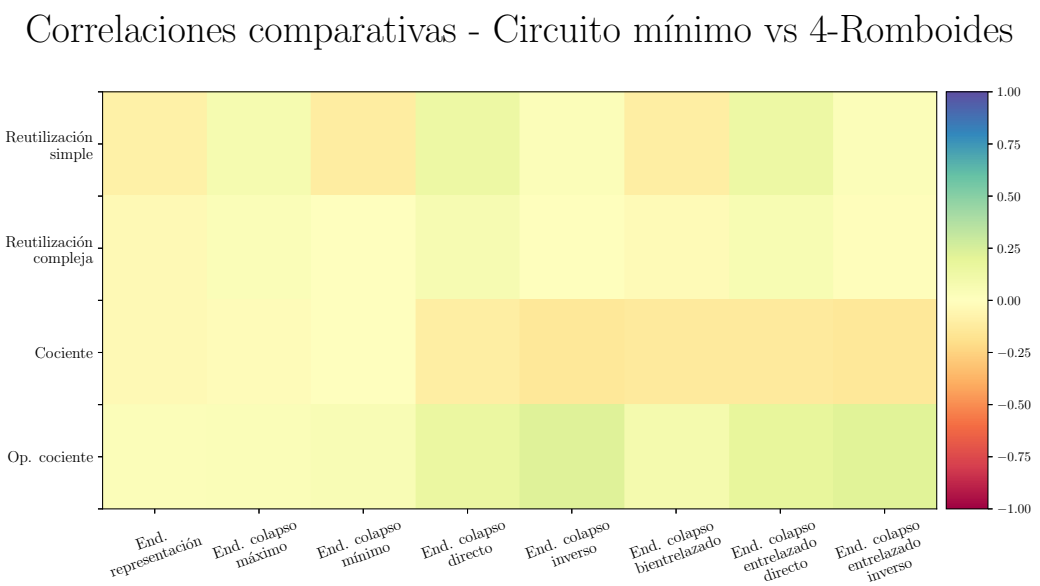


Figura 5.45: Circuitos mínimos frente a circuitos con forma 4-romboidal. Volver

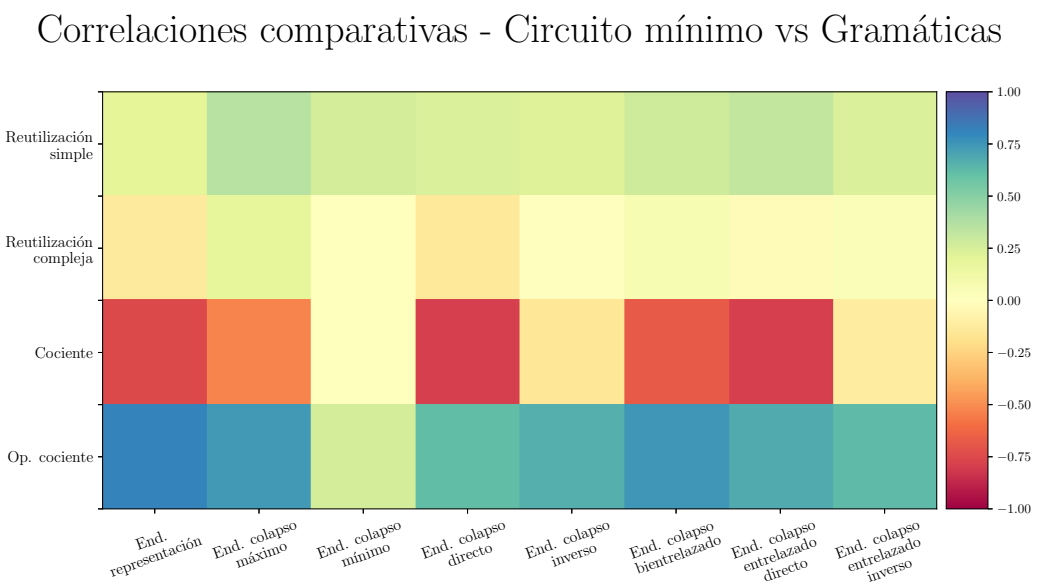


Figura 5.46: Circuitos mínimos frente a gramáticas. Volver

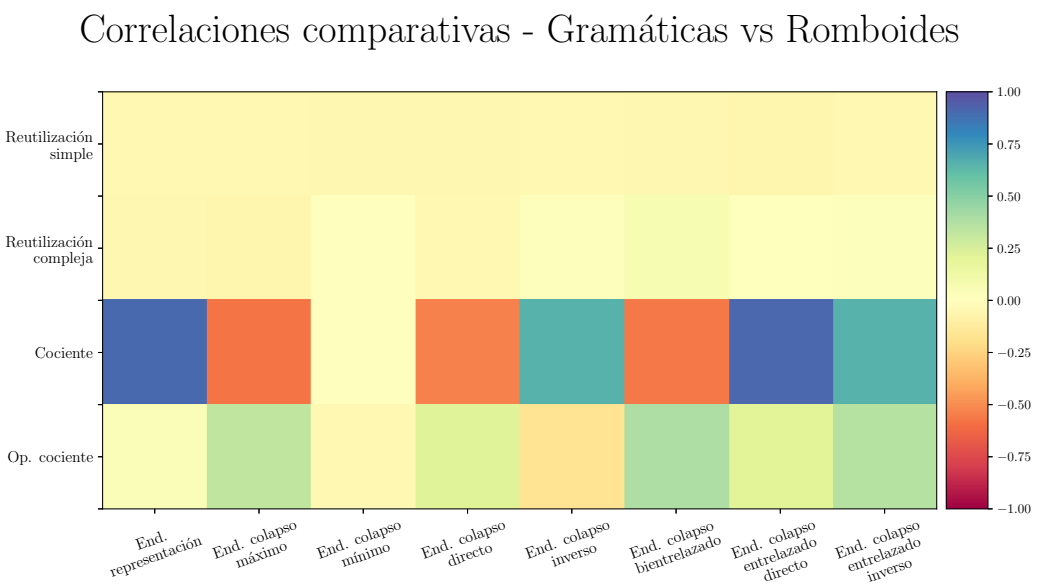


Figura 5.47: Gramáticas frente a circuitos con forma 4-romboidal. Volver

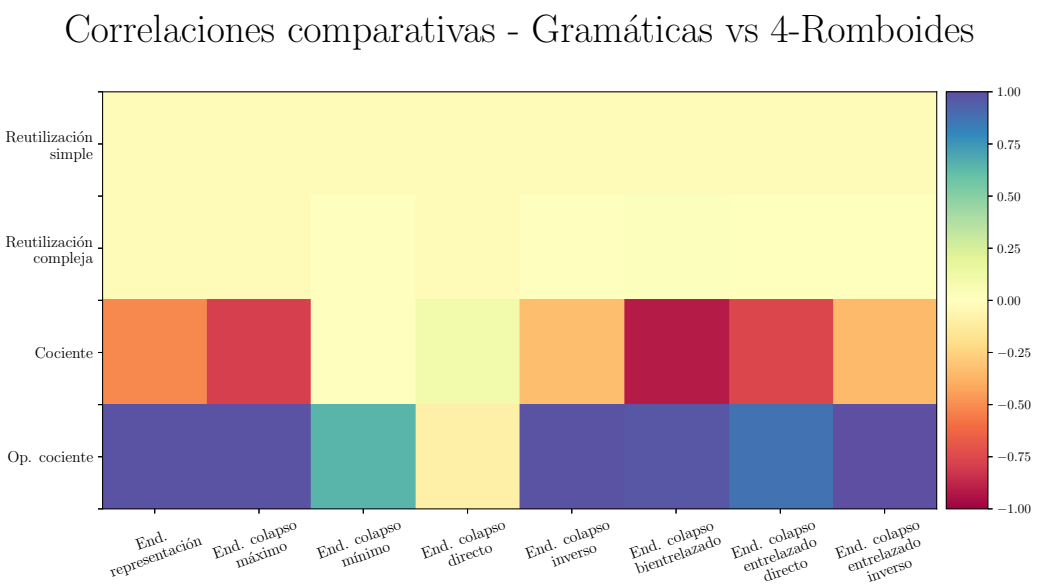


Figura 5.48: Gramáticas frente a circuitos con forma 4-romboidal. Volver

Bibliografía

[1] J. Baumgartner y A. Kuehlmann. *Min-area retiming on flexible circuit structures*. 2001. DOI: 10.1109/ICCAD.2001.968615.

[2] Sanjeev Arora y Boaz Barak. *Computational Complexity - A Modern Approach*. 2009.

[3] *Problema del milenio*. URL: https://www.claymath.org/millennium-problems.

[4] *Desigualdades para olimpiadas matemáticas*. URL: http://www.acm.ciens.ucv.ve/main/entrenamiento/material/desigual.pdf.

[5] Enrique Román Calvo (Galieve). *Algoritmo del índice (recorrido sistemático del espacio de circuitos booleanos)*. 2020. URL: https://github.com/Galieve/TFG-Informatica.

[6] *Punteros compartidos*. URL: http://www.cplusplus.com/reference/memory/shared_ptr/.

[7] *Herramientas de detección de acceso incorrecto a memoria*. URL: https://github.com/adah1972/nvwa.

[8] *Patrón threadpool*. URL: https://social.technet.microsoft.com/wiki/contents/articles/13245.thread-pool-design-pattern.aspx.

[9] *Threadpool para C++*. URL: https://github.com/vit-vit/CTPL.

[10] *Librería de enteros de precisión infinita*. URL: https://github.com/sercantutar/infint.

[11] *Software automake*. URL: https://www.gnu.org/software/automake/.

[12] *Librería pandas*. URL: https://pandas.pydata.org/.